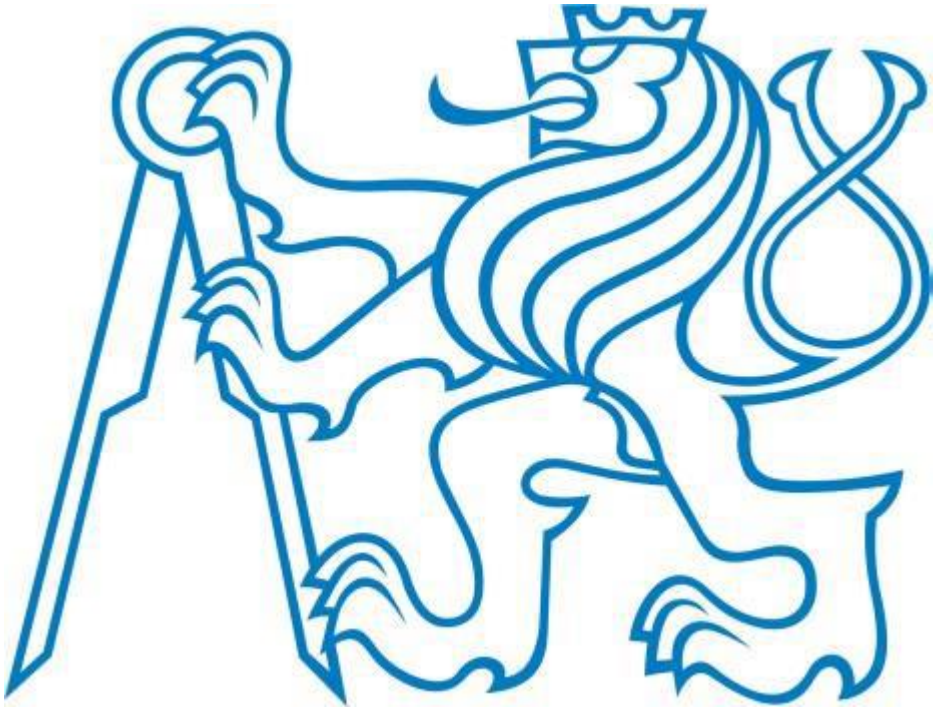
Czech Technical University in Prague

Faculty of Mechanical Engineering

Department of Automatic Control



Master Thesis

Title: Pedestrians Detection in Images for Autonomous Vehicles

Supervisor: Oswald Cyril

Yang Di

Prague – 2019

Master Thesis

I submit this thesis for review and defense in partial fulfillment of the requirements, for the degree master at Czech Technical University in Prague.

I declare that this dissertation is my own work, and all the sources have been quoted and acknowledged by means of complete references.

ACKNOWLEDGEMENTS

I would like to express my thanks to my supervisor Ing. Cyril Oswald, Ph.D., who gives me the valuable instructions, advices and supports during my study. Also for the lectures of Deep Learning, this gave me knowledge during my course.

I would like to give special thanks to my family and friends for encouragement, and patient waiting me when I study abroad here.

Yang Di

Prague, June, 2019

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Yang Di** Personal ID number: **473129**

Faculty / Institute: **Faculty of Mechanical Engineering**

Department / Institute: **Department of Instrumentation and Control Engineering**

Study program: **Mechanical Engineering**

Branch of study: **Instrumentation and Control Engineering**

II. Master's thesis details

Master's thesis title in English:

Pedestrians Detection in Images for Autonomous Vehicles

Master's thesis title in Czech:

Detekce chodců v obrazech pro potřeby autonomních vozidel

Guidelines:

1) Conduct the literature survey of appropriate neural networks architectures and optimization methods.
2) Create the appropriate data set for neural network training and validation.
3) Design the algorithm for pedestrians detection in images from created data set.
4) Validate your algorithm.

Bibliography / sources:

• Ian Goodfellow and Yoshua Bengio and Aaron Courville. "Deep Learning". MIT Press 2016.
• Benenson, Rodrigo, et al. "Ten Years of Pedestrian Detection, What Have We Learned?". ECCV 2014 Workshops. Springer International Publishing, 2014, p. 613-627.
• Bengio, Yoshua, A. Courville, and P. Vincent. "Representation learning: a review and new perspectives". IEEE Transactions on PAMI 35.8. 2013. p. 1798-1828.
• R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". CVPR, 2014.
• K. He, X. Zhang, S. Ren, and J. Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition". ECCV, 2014.

Name and workplace of master's thesis supervisor:

Ing. Cyril Oswald, Ph.D., U12110.3

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **26.04.2019** Deadline for master's thesis submission: **12.06.2019**

Assignment valid until: _____

Ing. Cyril Oswald, Ph.D. Supervisor's signature	Head of department's signature	prof. Ing. Michael Valášek, DrSc. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

ABSTRACT

With the rapid development of the world’s economy, the number of vehicles is constantly increasing. Due to the driver's subjective error, traffic accidents occur frequently, which seriously threatens the life and safety of pedestrians on the road. The emergence of autonomous vehicles will reduce traffic accidents caused by human factors. Pedestrian detection is the most important technology in the automatic driving systems because man's life is more precious than anything. Pedestrian detection method is a valuable and challenging topic in the field of computer vision because pedestrian with both rigid property and flexible property, whose appearance is easily affected by such factors as clothes, occlusion, scale, posture, and viewed angle. This master thesis processes a pedestrian detection method based on traditional method (HOG+SVM) and neural network method (faster-rcnn). After comparing the detection precision of these two methods, it is obviously to find the second one is better.

This thesis mainly carries out the following research：

1.Introduce the theory of traditional objective detection method (HOG+SVM) and deep learning method (rcnn, fast-rcnn, faster-rcnn).

2.Due to the limitation of space, the faster- RCNN is selected as the detector algorithm.

3.Training these two detection models and comparing their precious. Verifying the superiority of neural network detector.

Key words: autonomous vehicles, rcnn, fast-rcnn faster-rcnn, pedestrian detection, HOG+SVM, objective detection

List of figures

List of tables

List of acronyms

CNN	Convolution Neural Network
RCNN	Region Convolution Neural Network
SIFT	Scale Invariant Feature Transform
HOG	Histogram of Oriented Gradient
SVM	Support Vector Machine
NMS	Non-Maximum suppression
ROI	Region of Interesting
RPN	Region Process Network
IOU	Intersection Over Union
Fast-RCNN	Fast Region Convolution Neural Network
Faster-RCNN	Faster Region Convolution Neural Network
COV Layer	Convolution layer + Pooling layer + Relu layer

1. Introduction

1.1 Research background and significance

With the development and progress of science and technology, repetitive and boring tasks that used to be completed by a large number of people are gradually handed over to computers. As a subject based on image processing, and machine learning, computer vision is a rapidly developing research field in recent years. Its main task is to simulate people's visual ability and try to build an artificial intelligence system that can obtain "information" from images or multi-dimensional data. The establishment of artificial intelligence system for detecting whether there are pedestrians in the image or video is called pedestrian detection. If there are, find out the position coordinates. Pedestrian detection is the basis of research on pedestrian tracking, behavior analysis, pedestrian identification and so on. Good pedestrian detection algorithm can provide strong support for these studies[1]. The main applications of pedestrian detection in industry include vehicle assisted driving, intelligent video monitoring, pedestrian behavior analysis and so on. Pedestrian detection has also been applied in new fields such as aerial photography and victim rescue in recent years. There are still challenges and difficulties to cover, because the objects are easily affected by clothing, scale, shielding and perspective.

Similar to computer vision, machine learning is a subject that enables computers to simulate human beings. Specifically, machine learning is concentrating how computers simulate or realize human learning behaviors to acquire new knowledge or skills, and reorganize these knowledge structures to continuously improve their own performance. In the late 1980s, Shallow learning was invented and enabled the neural network to conduct statistical learning from a large number of training samples to discover statistical laws and predict unknown events. But because of the difficulty of theoretical analysis and model training, it is hard to develop. In 2006, Hinton, professor at the university of Toronto, published a thesis called Deep Learning in Science[2]. After that, deep learning has been rapidly developed and applied to a variety of industrial applications.

There are many main application scenarios of pedestrian detection in real life and production. The following are three examples:

(1). Surveillance cameras are becoming more common in various public places. As the number of surveillance cameras grows and their definition improves, the total amount of video they can capture is increasing at an ever-increasing rate. How to deal with this huge amount of video data is becoming an increasingly serious problem. In sharp contrast to the ever-expanding amount of data, the current video monitoring method is still relatively backward and almost all of video monitoring work is completed by human. Even although manual monitoring has the advantages of being flexible and able to deal with special situations compared with using a machine for monitoring, it also has the disadvantages of easily causing the monitor tied, missing important information making economic losses.

Using pedestrian detection technology, we can use the computer to automatically complete the work of video monitoring and detect every pedestrian in the video screen instead of human, then analyze its behavior and trajectory, timely find abnormal conditions and automatically alarm. This can reduce the cost of human monitoring and improve the detection accuracy and economic benefits of enterprises.

(2). Automobile plays an important role in modern society. However, with the economic benefits and living convenience provided by automobile, traffic accidents also become an important factor to cause personal and property losses. Many traffic accidents could be avoided if vehicles were given the ability to predict and deal with dangerous situations by themselves. Pedestrian detection based on video image data and corresponding countermeasures are obviously an important part of realizing this kind of vehicle assisted driving ability. At present, many companies and academic institutions made many relevant researches, such as Google, Tesla, MIT and baidu. With these researches getting deep, the demanding for vehicle assisted driving technology is becoming stronger and stronger, which has been a hot issue of common concern of academia and industry. As an important part of vehicle assisted driving technology, pedestrian detection algorithm has made some progress, but its ability to face complex scenes is still a problem.

(3). People are the most important part of the environment where machines are located. With the rapid development of intelligent robot technology, it has become one of the most meaningful and challenging subjects in modern engineering to endow intelligent machines with the ability to interact with people. In order to make an intelligent robot work like a normal human, the first task is to make it have the ability to perceive the surrounding environment[3]. Also, detecting and identifying human

beings are most important works to be done, because most of the objects the robots service are human beings. Pedestrian detection technology can give the robot the ability to view the human in the surrounding environment. Then, analyzing the human behavior and needs on this basis.

In fact, there are many application scenarios of pedestrian detection technology. With the continuous development of machine intelligence, as long as there are scenes in machines which interact with humans and provide services, pedestrian detection technology is required to realize its functions.

In this thesis, it is focused on the pedestrian detection in autonomous car system because it seems like the eyes of car and the most import safety part to prevent the human life.

1.2 Research status

Pedestrian detection technology has been developed for decades and has made great progress in both speed and accuracy. However, even the most advanced pedestrian detection algorithm still has a long distance between it can be put into industrial application.

With the rapid growth of industrial demand for pedestrian detection technology in recent years, the pedestrian detection technology has experienced rapid development since the HOG feature was proposed in 2005. The academic community mainly focuses on improving the detection accuracy. Every year, there are many articles related to pedestrian detection in top journals in the field of image processing and computer vision (such as PAMI, IJCV, etc.) and top conferences (such as CVPR, ICCVetc)[4].

From Viola's proposal of combining Haar feature and AdaBoost method in 2001, to Dalal's HOG in 2005, and DPM in 2008, it can be seen that each revolutionary research in pedestrian detection field is closely related to the feature[5]. These studies greatly improve the detection performance by finding new shallow structure that can extract the most distinguishing feature of pedestrians. For this reason, in order to improve the performance of the algorithm, researchers often spend a lot of time on selecting more suitable features. Although this kind of "feature engineering" has a significant effect, it relies on the prior knowledge and experience of human beings. Therefore, it not only consumes a lot of manpower resources, but also the algorithm

cannot learn features from data by itself. The purpose of deep learning is to enable machines to learn by themselves and making it easier to extract useful information for classifiers or predictors. The article "Learning Deep Architectures from AI" theoretically proves that the Deep network structure has the potential to achieve better performance than the shallow structure in solving complex problems (such as visual signal and audio signal processing). Therefore, deep learning is better than traditional methods in the field of pedestrian detection. That is why I choose deep learning as my research direction.

Krizhevsky proposed AlexNet became the milestone in the process of convolution neural network development. Since 2012, the depth of the learning algorithm has been widely used in the field of computer vision, therefore appeared GoogleNet, VGG16, ZF, such as new network structure[6].

It extended from the classifier (used for image classification and recognition, etc.) to detector (used for target detection, etc.) On the basis of RCNN, new algorithms such as Fast RCNN, SPPNet and Faster RCNN have appeared[7]. The performance of target detection has been continuously improved. Academics on new progress in the depth of the network makes a single category target (such as pedestrians) attention shifted to the general target detection, such as Fast RCNN 20 classes and class based on the 1000 AlexNet etc., and to the general target detection for this year. According to PASCAL VOC performance is getting better and better, Faster RCNN algorithm in general target detection task on the macro average accuracy has reached 73.2% (mAP)[8].

2. System design and feasibility analysis

2.1 Design requirements and indicators

1. Conduct the literature survey of appropriate neural networks architectures and optimization methods.

2. Create the appropriate data set for neural network training and validation.

3. Design the algorithm for pedestrians' detection in images from created data set.

4. Validate algorithm.

2.2 Work structure

Based on deep convolutional neural network and open source pedestrian dataset, this thesis studies the application of pedestrian detection algorithm:

1. Analyze the process and development trend of traditional detection algorithms, and compare the improvement of convolution neural network based detection algorithms in the detection framework, focusing on the detection algorithm of Faster RCNN.

2. Based on the analysis of the actual performance and results of the above open-source pedestrian data set of Faster RCNN, this thesis puts forward some improvement measures, including:

(1). Improving the basic network based on the experimental results on the basis of CaffeNet and VGG_CNN_M_1024.

(2). Using IOU with different thresholds for comparison of predictions.

(3). In the aspect of precision and recall rate compared with traditional target detection algorithm.

(4). Using datasets, INRIA, to make fine-tuning, to further improve the algorithm.

(5). Completing detection model training tasks at the same time on the test set, the improved algorithm with the original algorithm, and increase the contrast experiment of the algorithm in different improvement measures.

(6). Analysis of the experimental results and summarize the deficiencies what as the main content of the next step research work.

3.Theory of algorithms

3.1 Traditional objective detection method

In the early stage, researchers usually defined a series of mathematical operations as the process of feature extraction, namely, manual feature extraction operator. In order to resist image noise, illumination change, target scale change, attitude change and other adverse factors, such algorithms usually need to use image scale space, feature multi-direction, Gaussian kernel, gradient statistical histogram and other methods. Using deep convolution neural network to extract the characteristics of quality has been far more than the former algorithm. A large number of experimental results also show that the depth of the convolution of the neural network characteristics of high-level semantic information is stronger than the manual operator of feature extraction of semantic information. In this section, firstly, SIFT and HOG are two typical manual feature extraction operators to explain their algorithm principles in detail[10]. Then, the structure and classification principle of deep convolutional neural network, as well as the historical development process and the latest progress of deep convolutional neural network are emphatically introduced.

3.1.1 Manual feature extraction operator

As mentioned above, SIFT feature extraction operator is a typical representative of early manual feature extraction operator, and its algorithm flow is as follows:

(1). construct scale space

The same objects in different scales of observation will also change, for example, in the same scene captured a blurring image vision and a picture, although the two have some common object in the image, but the pixels in the images of the difference between the two will be relatively large in details, the characteristics of the scale of the robustness requirements will be higher. Therefore, constructing scale space is a necessary step.

It is a natural idea to construct multi-scale sequence of images first, among which spatial pyramid is a common method to construct multi-scale sequence of images, as shown in figure 1. On the basis of this multi-scale strategy, Gaussian kernel and pixel gradient below the second order and dimension space theory gradually become an important part of scale space. Scale space expression is also using the semigroup

properties of Gaussian kernel, namely two Gaussian convolutions is equal to a parameter of the Gaussian convolution, signal representation not only makes the coarse scales can be directly from the original signal and Gaussian convolution, can also from fine scale signal expression and high nuclear convolution, enrich the way of feature extraction[11].

FIG1. IMAGE PYRAMID DIAGRAM[11]

It is proved that the kernel other than Gaussian kernel will affect the image in addition to blur. Therefore, SIFT uses a two-dimensional Gaussian kernel function to make the image smooth and eliminate other changes in the image besides blurring. Suppose a single-channel image, SIFT uses formula (1), (2) to construct the scale space of the image.

$$G(x,y,z) = \frac{e^{-\frac{x^2+y^2}{2z^2}}}{2\pi z^2} \quad (1)$$

$$L(x,y,z) = G(x,y,z) * I(x,y) \quad (2)$$

Where, $I(x,y)$ represents the image after descending sampling or the original image, G (x, y, z) represents the Gaussian kernel function, and $L(x,y,z)$ represents the feature obtained when the ruler is used. The size of the image determines the smoothness, which corresponds to the different blurring degree of the image. In order to reduce the amount of computation, SIFT reduces the sampling process of $L(x,y,z)$ as the input image of the next scale, which needs to be increased correspondingly.

(2). Key point extraction

In order to get the features of scale invariance, SIFT uses the second-order Laplacian operator to solve the extremum point for the scale space generated above,

but the direct second-order Laplacian operator processing is of great computational complexity, SIFT uses approximate processing, such as formulas follow:

$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} \tag{3}$$

$$G(x, y, kz) - G(x, y, z) \approx (k-1)z^2 \Delta^2 G \tag{4}$$

$$D(x, y, z) = |G(x, y, kz) - G(x, y, z)| * I(x, y) \tag{5}$$

$$D(x, y, z) = (k-1)z^2 |\nabla^2 G| * I(x, y) \tag{6}$$

Where, $\nabla^2 G$ is the second Laplace operator, and z is the scale.

Since the image is discrete, SIFT compares the Gaussian difference operator of each pixel with the pixel points in its 8 neighboring regions and the pixel points in its 9 neighboring regions with a total of 26 pixels in the two adjacent scales to get the maximum value, which is also the step that consumes the most operation time[11].

The extremum obtained in discrete space is not necessarily the true extremum of continuous function. Therefore, SIFT performs curve fitting on Gaussian difference operator to remove the points with extremely asymmetric local curvature, which are mainly low contrast points and unstable edge response points[12]. The formula corresponding to the contrast of key points is shown in formula (7), (8):

$$D(\hat{l}) = D(l) + \frac{1}{2}\frac{\partial D(l)^T}{\partial l} \tag{7}$$

$$\hat{l} = \frac{\partial^2 D^{-1}}{\partial l^2}\frac{\partial D}{\partial l} \tag{8}$$

Where, l represents the key points. For the key points where $D(l)$ exceeds the threshold, if less than the threshold, the contrast ratio is too small to effectively classify. The curvature of key points is calculated by formula (9-12)

$$\mathbb{Z} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \tag{9}$$

$$\alpha, \beta = \lambda(H) \tag{10}$$

$$\alpha = \gamma\beta \tag{11}$$

$$\frac{T(H)^2}{D(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(\gamma+1)^2}{\gamma} \tag{12}$$

Where $\mathbb{Z}$ represents the second-order Hessian matrix, λ (*) represents the eigenvalue operation, T (*) represents the trace of $\mathbb{Z}$, D (*) represents the determinant of $\mathbb{Z}$, and $T(H)^2/D(H)$ represents the principal curvature. Key points where the main curvature exceeds the threshold are discarded, and the remaining extreme points are SIFT key points [13].

3. Solve the main direction of the key point

To achieve robustness of feature points to image rotation, SIFT assigns direction parameters such as formula (13,14) to each key point

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{13}$$

$$o(x,y) = \tanh^{-1}\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)} \tag{14}$$

By combining the above three calculation formulas, the location, scale and direction of the key points can be obtained.

4. Key point descriptor generation

Take the key points as the center, calculate the pixel gradient of the surrounding 16x16 pixel region, and divide each quadrant into 4x4 sub-pixel region. In the statistical gradient histogram, SIFT counts 45 degrees as a column from 0 to 360 degrees, so there is a total of 8 columns, that is, each sub-region has an 8-dimensional direction vector. Eventually, SIFT will generate 128-dimensional feature vector descriptors for each key point. The 128 - dimensional feature vector descriptor is SIFT extracted features[13]. HOG is a classical feature extraction operator proposed for pedestrian target. Compared with SIFT, HOG greatly improves the accuracy in the pedestrian data set of MIT, and also opens up a new idea to extract the features of pedestrian targets. The algorithm flow is as follows:

(1). Image normalization

First, convert the input image into a single channel image, ignoring the color information; Secondly, in order to reduce the influence of illumination factor, HOG uses Gamma compression formula, such as formula (15)

$$H(x,y) = I(x,y)^2, g \in (0,1) \tag{15}$$

(2). Calculate global gradient

Like SIFT, HOG also needs to calculate the gradient value of pixel, as shown in formula (16-19)

$$G_x(x,y) = H(x+1,y) - H(x-1,y) \tag{16}$$

$$G_y(x,y) = H(x,y+1) - H(x,y-1) \tag{17}$$

$$m(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{18}$$

$$o(x,y) = \tanh^{-1}\frac{G_x(x,y)}{G_y(x,y)} \tag{19}$$

Where m (x, y) represents the length of the gradient, and o (x, y) represents the direction of the gradient.

(3). Cell units were segmented and histograms were calculated

HOG divides the normalized image into several sub-regions, namely, cell units, such as pixel regions with the size of 8x8. HOG calculates the histogram of gradient distribution in a column interval of 20 degrees for each cell unit, with statistical ranges of 0~180 degrees and 181~360 degrees respectively. Since the pixel gradient value of the same interval is different, HOG takes the gradient value as the weight of the statistical times.

(4). Cell units are grouped into blocks

When designing, HOG follows the idea from pixel to local to global. Therefore, based on cell unit, HOG strings cell units of a certain size, such as 2x2 cell units, into a block in a random order[14]. The combined blocks then take the length and width of cell units as step sizes to conduct sliding window sampling in horizontal and vertical directions, and finally form the feature description of the whole picture. Obviously, window overlap will occur in the process of sampling, so the global normalization of features is needed finally.

3.2 Neural network detection method

3.2.1 Theory of CNN

3.2.1.1 CNN introduction

CNN can choose window (Proposal) feature extraction and classification. It comes from the traditional full connection Neural Network model development and to the depth of the Network model, that is one of the important algorithm is widely used in computer vision field. The main feature of CNN is that it draws on the concept of human visual perception field and proposes the concept of weight sharing[15].

The traditional neural network model is fully connected, that is, each neuron in each layer of the network is connected to all neurons in the upper and lower layers, so if the number of neurons in the two adjacent layers are M and N respectively, then only these two neurons are connected.

The weight number of connections between the five layers is M*N. When the network depth increases, the number of parameters to be learned in the network increases rapidly, which is difficult to realize. As shown in the figure below, if the input an image with 1000 * 1000 pixel, the number of neurons and neural network in the first layer and the image pixel number is equal to 1 million, then under the condition of full connection (each neuron is connected to each pixel value), the network needs to learn the parameters of the number is $1000 * 1000 * 1000000 = 10^{12}$, which is 10 to 12 weight parameters. This is undoubtedly a very large number, and the enormous computing resources and time required to train the network make it difficult to apply in practical applications.

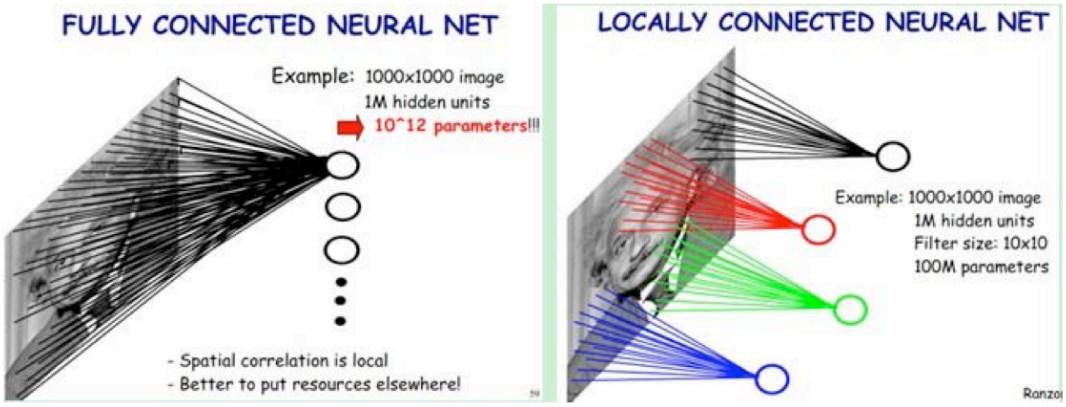


FIG.2. SCHEMATIC DIAGRAM OF FULLY CONNECTED NEURAL NETWORK AND LOCALLY CONNECTED NEURAL NETWORK. THE LEFT FIGURE SHOWS THE FULLY CONNECTED NEURAL NETWORK, AND EACH NODE IN THE ADJACENT TWO LAYERS IS CONNECTED TO EACH OTHER. THE RIGHT FIGURE SHOWS THE PARTIAL CONNECTION OF THE NEURAL NETWORK, AND THE NODES OF THE NEXT LAYER ARE ONLY CONNECTED WITH SOME OF THE UPPER NODES[16].

However, CNN uses the concept of receptive field for reference. Each neuron does not need to feel the global information of the whole image, only local image information can be felt, and the information obtained by the neurons at the lower level is integrated on the neurons at a higher level, which greatly reduces the parameters that the whole network needs to learn. As shown in the figure 2, if we use 1 million neurons, each neuron only feels the region of 10*10 pixels (i.e., only convolving the 100 pixels), so the number of parameters to be learned in the network is reduced to 10^8.

On the other hand, CNN points out the concept of weight sharing. It is not hard to imagine that when we look at an image, no matter where we look in the image, the function of a particular neuron in our brain is to extract the same feature, that is, it will

not change the convolution kernel of 100 pixel values of different regions. Therefore, when extracting a feature, the convolution carried out on the whole image uses the same convolution kernel. Only 100 weights are needed for a feature to be expressed, the weights used to extract the same feature at different positions are Shared. So, we need as many convolution kernels as we need to extract as many features from an image. For example, we need to extract 1000 features in the field of 10*10, so we need 100 convolution kernels, Therefore, our neural network only needs to learn 10^4 weights in total, which is only 1/10,000,000 of 10^{12} parameters that the ordinary neural network needs to learn. In fact, in general we only need a few dozen convolution kernels, which is enough to represent various features.

With the concept of receptive field and weight sharing, CNN has made multi-layer artificial neural network possible for the first time, and it has been widely used in the field of image recognition. Traditional image features requiring manual design can be automatically learned and extracted by CNN. This data-driven learning method greatly simplifies the learning method of features and reduces the workload and difficulty of manual work. Now it has become the mainstream way of image semantic understanding in academia and industry[17].

3.2.1.2 Advantages of CNN

CNN is mainly used to identify displacement, scaling and other forms of distortion invariant two-dimensional graphics. Since the feature detection layer of CNN learns from training data, it avoids explicit feature extraction while learning implicitly from training data when using CNN. Moreover, since the weights of neurons on the same characteristic mapping surface are the same, the network can learn in parallel, which is also a major advantage of the convolutional network over the neural network connected with each other. Convolution weights of neural network with its local shared special structure in terms of speech recognition and image processing has its unique superiority, its layout is closer to real biological neural networks, a weight sharing reduces the complexity of the network, especially the multidimensional network input vector image can directly input this feature to avoid the data in the process of feature extraction and classification the complexity of the reconstruction. The existing classification methods of non-deep learning are almost all based on statistical features, which means that some features must be extracted before discrimination. However, explicit feature extraction

is not easy and is not always reliable in some application problems. Convolutional neural networks avoid explicit feature sampling and implicitly learn from training data. This makes the convolutional neural network obviously different from other classifiers based on the neural network. The feature extraction function is integrated into the multi-layer network through structure reorganization and weight reduction. It can deal directly with grayscale image and can deal directly with image-based classification[18].

Compared with general neural networks, convolutional neural networks have the following advantages in image processing:

(1). The input image will match the network topology well.

(2). Feature extraction and pattern classification are carried out at the same time and are generated in training.

(3). Weight sharing can reduce the training parameters of the network and make the structure of the neural network simpler and more adaptable.

3.2.2 Theory of RCNN

3.2.2.1 RCNN introduction

RCNN is a target detection algorithm developed on the basis of CNN. The original convolutional neural network is a classifier, which can only classify an input image and cannot detect the specific position of the target on an image that is not just one target. The most important contribution of RCNN is to extend the application of convolutional neural network from the classifier to the detector. It first conducts selective search on an input image and produces about 2000 candidate regions, and then uses CNN network to extract features and classify each candidate region to determine whether it is a target to be tested.

The RCNN algorithm is divided into four steps:

(1). An image generates 1000~2000 candidate regions.

(2). For each candidate region, deep network is used to extract features.

(3). Features are sent to SVM classifiers to each class that determine whether they belong to this class.

(4). fine correction of candidate box positions using regressions[19].

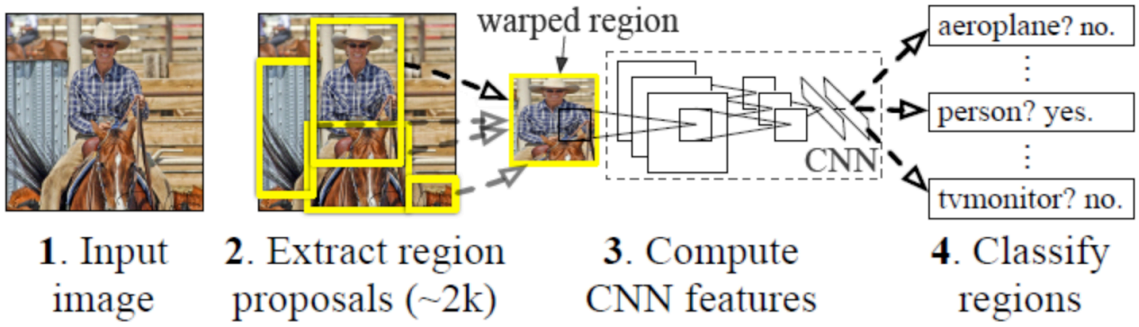


FIG. 3. STRUCTURE OF RCNN[20]

3.2.2.2 Means of RCNN

RCNN make the CNN method into the field of target detection, which greatly improves the effect of target detection and can be said to change the main research ideas in the field of target detection.

Compared to the traditional method, RCNN has the following advantages:

(1). Speed: the classical target detection algorithm uses sliding window method to judge all possible regions successively. In this paper[21], the selective search method is used to extract a series of candidate regions for objects in advance, and then only extract features on these candidate regions (CNN) for judgment.

(2). Training set: classical target detection algorithm extracts manually set features in the region. From RCNN was born, deep network is used for feature extraction. Use two databases: a large recognition library is calibrating the category of objects in each image. Ten million images, one thousand categories. A smaller inspection library is position of objects in each image, 10,000 images, 20 classes. In this paper[22], the identification library is used for pre-training to obtain CNN (supervised pre-training), and then the detection library is used for tuning parameters, and finally the evaluation is carried out on the detection library.

3.2.2.3 Disadvantage of RCNN

Although RCNN has pioneered the neural network target detection, it still has some shortcomings：

(1). It can be clearly felt that its computational load is very large. After all, feature calculation should be carried out for each candidate region.

(2). Too much redundant calculation. After all, candidate areas are highly overlapped.

(3). It's not end-to-end training. It's a hassle.

(4). Memory footprint: need to store multiple SVM classifier and bounding box regression.

(5). There are rigid requirements on the size of input pictures.

3.2.3 Theory of Fast-RCNN

3.2.3.1 Fast-RCNN introduction

Because RCNN has many disadvantages, Fast-RCNN was born. Fast-RCNN is on the basis of the RCNN was improved, first on the depth of the whole image using volume product network computing feature maps (feature map), then the selective search to get candidate area on the characteristic graph corresponding area, again to this area to use two full connection layer to generate a feature vector, and then input feature vectors into to two separate independent full connection layer, one of the classified using softmax, another to get the position of the bounding box of regression to obtain more accurate location information[23]. In this way, the operation time of using network forward propagation for each candidate region is saved, and only one forward propagation is needed for a whole image, which greatly accelerates the operation speed.

3.2.3.2 RCNN process

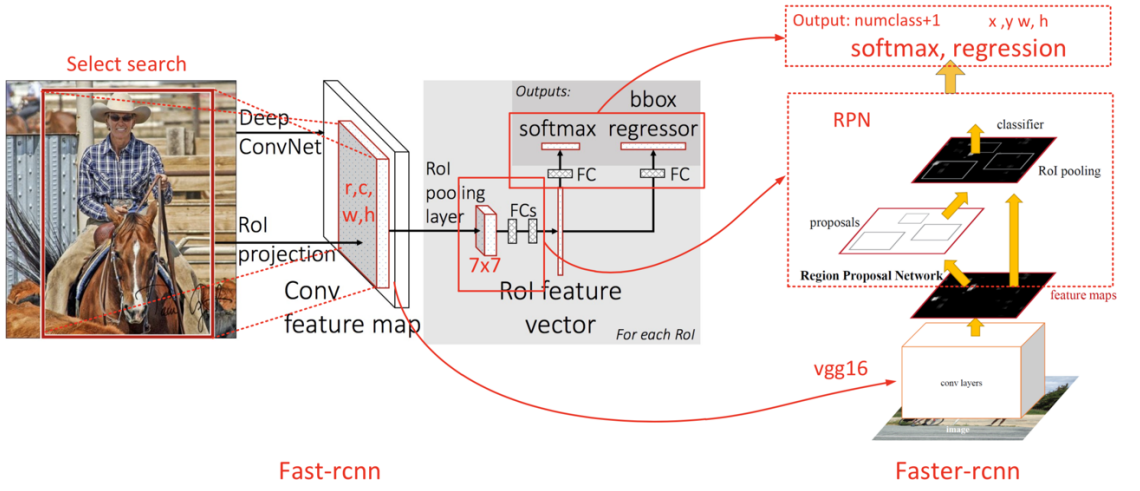


FIG.4. STRUCTURE OF FAST-RCNN[24]

Its process is as follows:

(1). Input pictures of any size into CNN to get the feature graph. In RCNN, there are 20 proposals for region proposals, which are equivalent to multiple convolution and waste of time.

(2). For the original image, the selective search algorithm is used to obtain approximately 2000 region proposals (equivalent to the first step of RCNN).

(3). In the feature map, find the corresponding feature box for each region proposals. Pool each feature to a uniform size in the ROI pooling layer.

(4). Fixed size feature vectors are obtained from uniform size feature box through full connection layer, and softmax classification (softmax was used to replace multiple SVM classifiers in RCNN) and bbox regression were conducted respectively.

Fast RCNN has two output layers, namely, classified score and regional position,.The loss function of the network in training also needs to take into account the losses in the two aspects respectively. It is assumed that the probability distribution value of K+1 class of each ROI is p= (p 0..., p K), the position of ROI is $\overrightarrow{t^k} = (x, t_y^k, t_w^k, t_h^k)$.

ROI was marked with category u and position v of ground truth, so the loss function during training was

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u > 1]L_{loc}(t^u, v) \qquad (20)$$

$L_{cls}(p, u) = -\log(p_u)$, is the logarithmic loss function for the actual class u.

3.2.4 Theory of Faster-RCNN

3.2.4.1 Faster-RCNN introduction

After RCNN and Fast-RCNN accumulation, Ross b. Girshick in 2016, put forward a new Faster-RCNN that comprehensive performance has improved greatly and the increasing of detection speed is particularly obvious[24].

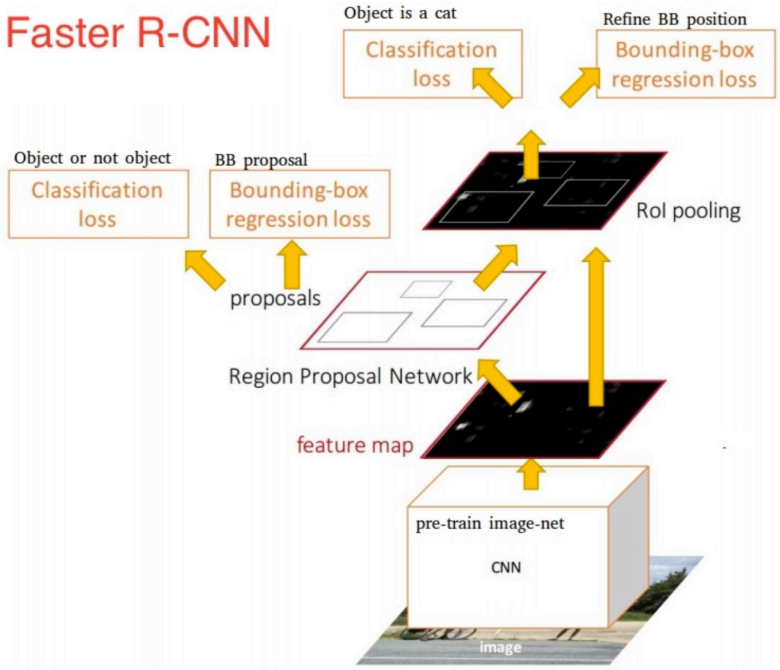


FIG.5. STRUCTURE OF FASTER-RCNN[25]

As we can see the figure 5, it is the overall of Faster-RCNN process:

(1). Convolution the layers. As a CNN network target detection method, Faster RCNN first uses a set of basic conv+relu+pooling layer to extract feature maps of image. The feature maps are shared for subsequent RPN and full connection layers.

(2). Region Proposal Networks. RPN network is used to generate region proposals. This layer by judging softmax anchors belongs to the foreground or background, using the bounding box regression fixed anchors to obtain precise proposals.

(3). RoI Pooling. Proposals feature maps and proposals are submitted for proposals in this layer. After synthesizing these information, the proposal feature maps are

extracted and sent to the following full connection layer to determine the target category.

(4). Classification. Using proposal feature maps calculation proposal category, at the same time, bounding box regression for testing box again the precise location

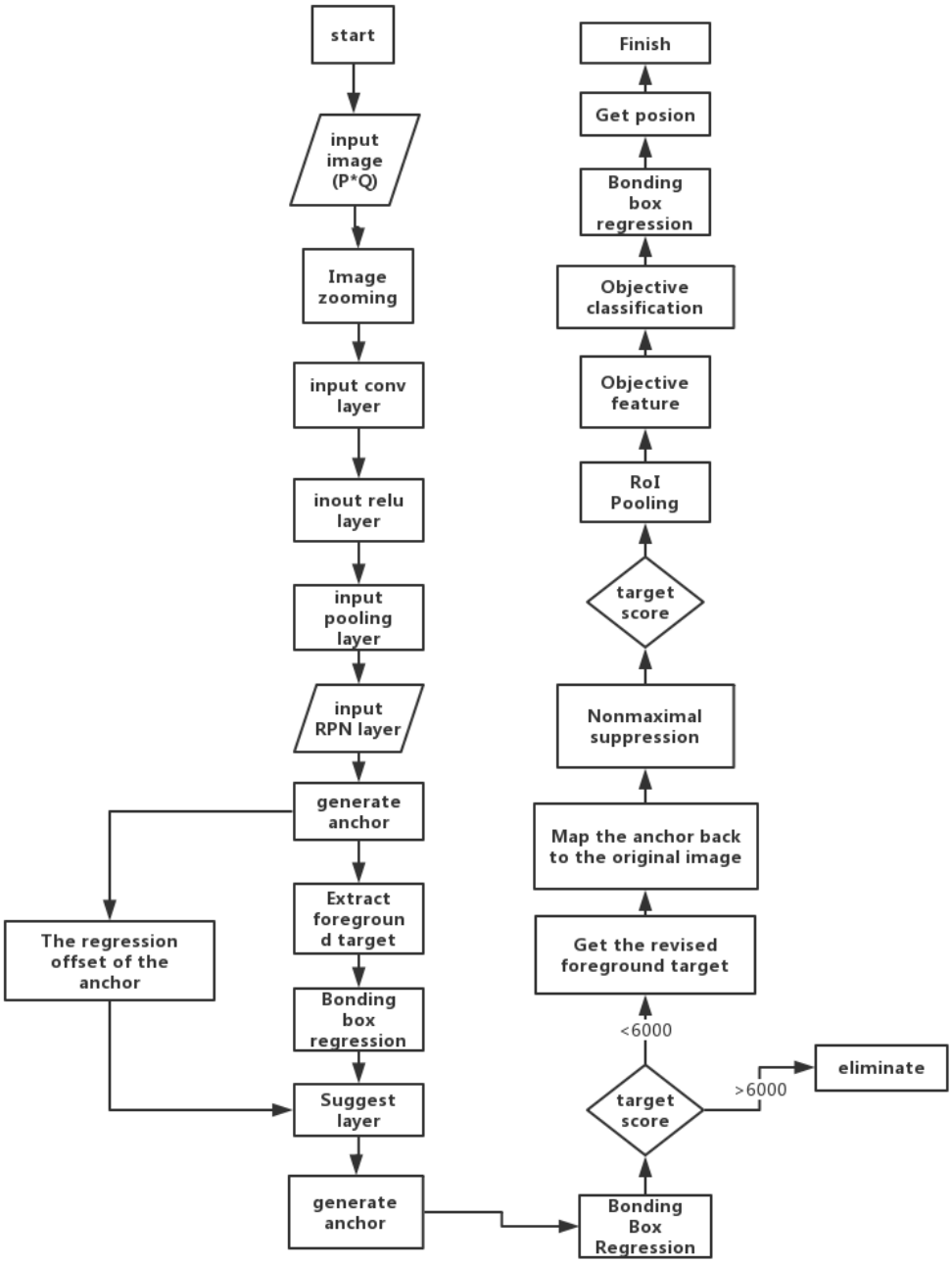


FIG.6. THE DATA PROCESS OF FASTER -RCNN

3.2.4.2 CONV layer

CONV layers include three layers: Convolution, pooling and relu. Here is a very easy to ignore but extremely important information which are used in the full CONV layer:

1. All of convolution layers are kernel_size = 3, pad = 1, stride = 1.

2. All of pooling layers are kernel_size = 2, pad = 0, stride = 2.

In Faster-RCNN CONV layers, all the convolution was processed by edge broadening (pad=1, that is, a circle of 0 was filled), resulting in the original image becoming (M+2)*(N+2) size, and then output M*N after 3x3 convolution. It's this setup that causes the size of matrix is not changed from input to output. As shown in figure 7.

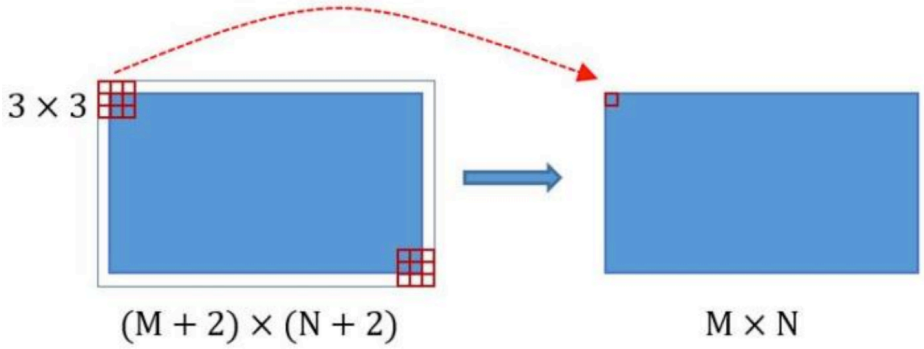


FIG.7.PROCESS OF MOVING CONVOLUTION[26]

Similarly, kernel_size=2 and stride=2 for the pooling layer in Conv layers. In this way, the M*N matrix for each pooling layer will be changed to (M/2) *(N/2) size. To sum up, in the whole Conv layers, Conv and relu layers do not change the size of input and output, and only the pooling layer changes the output length and width to 1/2 of the input.

Then, a matrix of M*N size is fixed to (M/16) *(N/16) after CONV layers. Thus, the feature map generated by CONV layers can correspond to the original map.

3.2.4.3 Region Proposal Networks introduction

The huge advantage of Faster-RCNN mainly lies in the design of RPN. The traditional selective search method is time-consuming to generate detection boxes. RPN is much faster.

The role of RPN is to extract candidate boxes, which is similar to the first step of Selective Search for RCNN. Its network structure is based on neural network, but the output is a multitask model including binary softmax and bbox (bounding box) regression. The input of the RPN network is the feature maps of the CNN output above. We do a sliding window operation with a size of 3*3 convolution kernel on the feature map and get a feature graph with a dimension of 256, the size of which is the same as the feature graph of input, and the dimension is 256*H*W. For this 256-dimensional vector, we will do 1*1 convolution operations twice, one to get 2000 score and one to get 4000 coordinates. This 2000 score only distinguishes whether the target is a target, and the score that the output candidate region belongs to the foreground (object) and background. Here, note that the classification here only distinguishes whether the target is included, and the category of the included target is what the final classification network of Faster-RCNN does. 4000 coordinates refer to a deviation from the original coordinates.

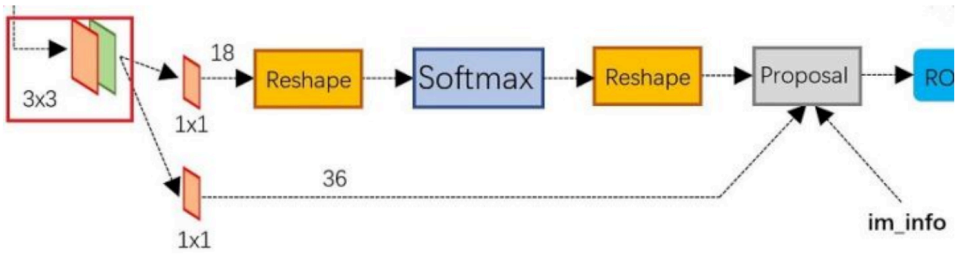


FIG.8.PROCESS OF RPN WORK[27]

Figure 8 shows the specific structure of the RPN network. Can be divided into 2 lines, see the RPN network actual above a foreground and background were obtained through the softmax classification anchors (foreground detection target), the following one is used to calculate for anchors the bounding box of regression offsets, in order to obtain accurate proposal. While the final Proposal layer is responsible for the comprehensive foreground anchors and bounding box regression offset for proposals, proposals and eliminate small and beyond borders. In fact, when the whole network reaches the Proposal Layer, it has completed the function equivalent to target positioning[27].

3.2.4.4 RoI pooling

Since the proposal is on the scale of M*N, spatial_scale is used to map it back to the size of (M/16) *(N/16) feature map. The regional level of the feature map corresponding to each proposal was divided into a grid of {pooled_w} *{pooled_h}. Max pooling processing is conducted for each part of the grid.

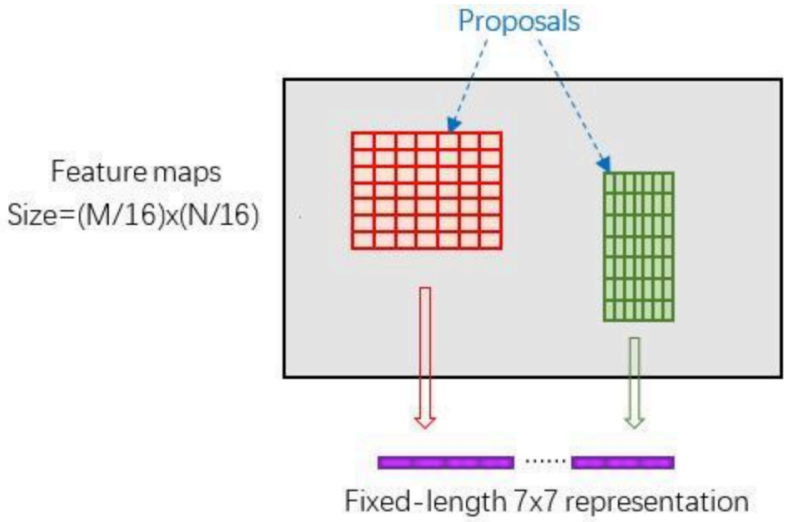


FIG.9. PROPOSAL PROCESS[28]

3.2.4.5 Classification

In the Classification part, the existing proposal feature maps are used to calculate which category each proposal belongs to (such as people, cars, TV, etc.) through the full connect layer and softmax, and the cls_prob probability vector is output. Again, at the same time using the bounding box regression for each proposal location offset bbox_pred, for return to a more precise target detection. The Classification part of the network structure is shown in figure 10.

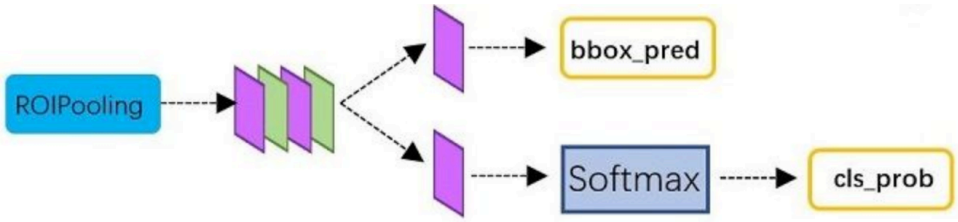


FIG. 10 CLASSIFICATION NETWORK STRUCTURE[29]

After the 7*7= 49-size proposal feature maps obtained from ROI Pooling are sent to the follow-up network, the following two things can be seen:

(1). for this part, there are 30 proposals for each proposal. There are 20 proposals for each proposal.

(2). The proposals for bounding box regression again, to get higher accuracy of the rectangle box.

4. Model training

This chapter will focus on the process of neural network training. As for the method of model training, there are two ways:

1. Alternative training (alt-op)
2. Approximate joint training (end to end)

This thesis uses the second one, because the speed of training is faster than alt-op, and save memory compared to first one. Meanwhile, as for the model choosing, this thesis takes the VGG_CNN_M_1024 to training, which Faster-RCNN provide the Caffenet model, VGG_CNN_M_1024 model and VGG16 model.

As mentioned earlier, Faster-RCNN can be divided to RPN and Faster-RCNN. Even though there are two parts, but they both have a part from pre-training model except they have their own special part. In general, the training principles of the two are mainly divided into the following steps:

(1). Use the model to initialize the RPN, and train the RPN after the initialization. After the training, the model and the unique structure of the RPN will be updated.

(2). Similar to the previous step, the same model is used to initialize the Faster-RCNN network. Since the RPN has been trained, the proposed value is obtained by using the RPN that has been trained in the previous step. After the training, both the model and the Faster-RCNN network structure will be updated. It should be noted here that although the training of RPN and Faster-RCNN both use the same model, the training is carried out separately from each other, so the model generated after training is completely different, so the model after training of the latter is still not shared.

(3). The basic idea of this step and the second step is the same, but just the opposite. This time, the model trained in step 2 is used to initialize RPN, and the second training of RPN is conducted after that. However, the model will be locked this time, and the model will not be modified during the whole training process, while the RPN will continue to be updated after this training. Since RPN and Faster-RCNN adopted the same model in the training of this step, it was called a Shared model.

(4). In the last step, the Faster-RCNN training was performed for the Faster-RCNN training model. The joint network of RPN and Faster-RCNN has been established.

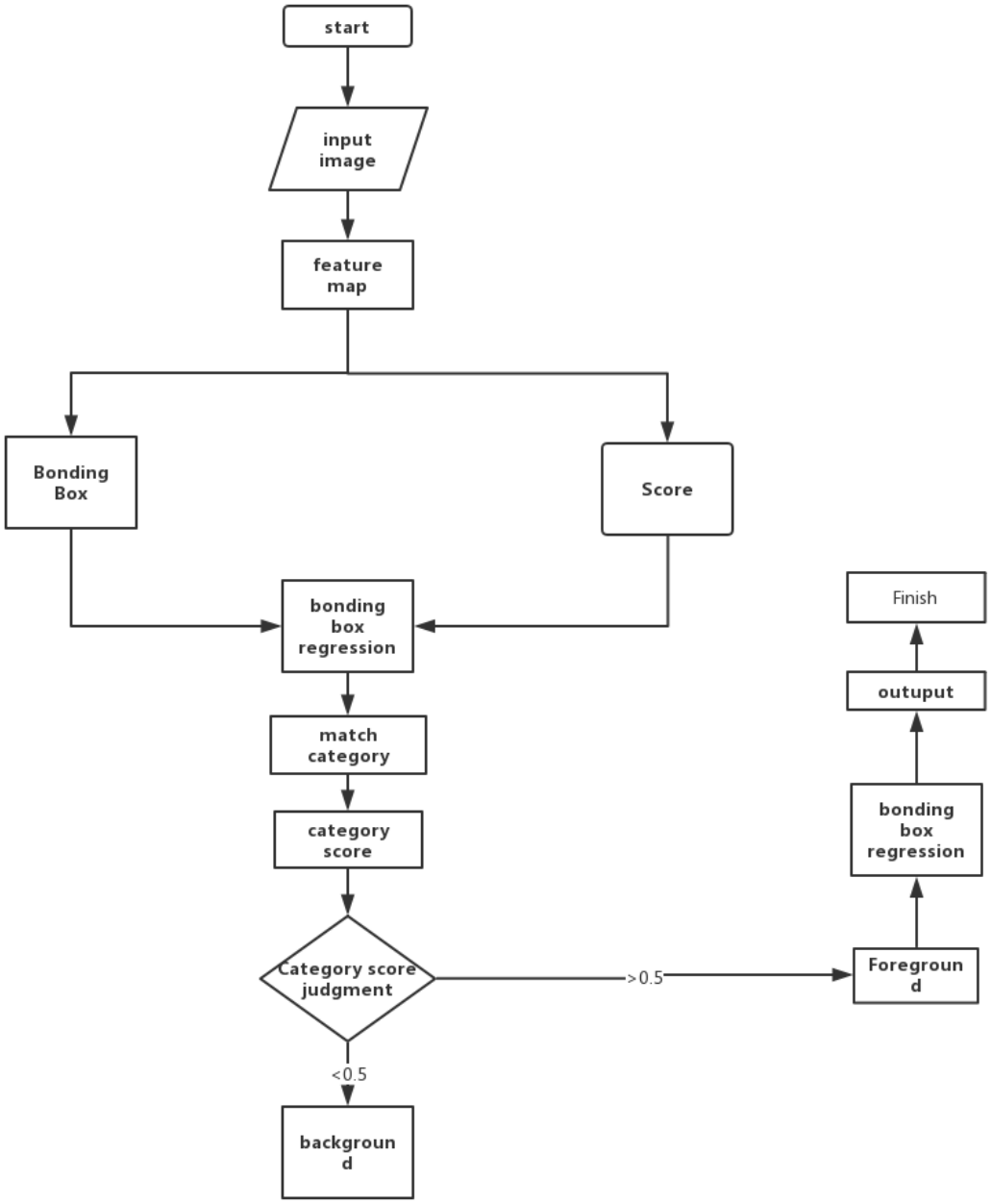


FIG.11. ARCHITECTURE OF FASTER-RCNN ALGORITHM

Figure 11 shows the overall flow chart of the algorithm. For each input image, it is first input into the CNN network (convolution layer) for feature extraction to generate the convolution feature map. Then RPN uses the generated convolution feature graph to generate anchors on the image. Meanwhile, output the score of target (only care about if it is the objection, ignore another objection. It is aim to get rid of the bad consequence). Then, take the bounding box regression to adjust the anchors. In this

case, the reason of using RPN is easier than normal convolution network. Therefore, it will be faster in training and reduce the time of running.

On top of that anchors, the rest of work is to divide these anchors to own classification and match them. After the ROI pooling layer, we arrive the last step of Faster-RCNN. Matching the feature with the objection though the faster-RCNN. It orders to build score on the anchors and adjust the position of bounding boxes[30].

4.1 Caffe introduction

Caffe is a neural-network framework develop by BAIR for implementing neural-networks to research application fast and easy. It readily offers model definitions, optimization settings and pre-trained weights.

This thesis uses caffe as the deep learning framework. The process of training and testing neural network model are both in the caffe.

4.2 Principle of neural network training

4.2.1 Training files framework

There are several folders in the project directory:

(1). caffe-fast-rcnn: Caffe framework storage directory

(2). data: pre-trained model storage directory and read the file's cache

(3). experiment： For storing configuration files and running log files, in addition, scripts can be used in end to end or alt-opt two training modes in this directory.

(4). Lib: store some python I/O files and their sub files

(5). Datasets: Mainly responsible for database reading

(6). Fast_rcnn: Mainly store python training, test code and training file config.py

(7). Nms: No maximal suppression

(8). Roi_data-layer: ROI operation

(9). RPN: code of RPN and define the method of anchor generation.

(10). Models: there are two models: Caffenet model, VGG_CNN_M_1024 model

(11). Output: This is the output directory after training, which will be stored in the faster_rcnn_end2end folder by default

(12). Tools: this is the training and test python files.

4.2.2 Training environment

1. The hardware configuration: 16G RAM, Intel i7-8750H CPU, NVIDIA GTX_1070 graphic card and Ubuntu 14.04.6LTS operating system.

2.The software configuration: caffe, pycaffe

3. Install py-faster-rcnn: faster-RCNN is open source. It has python version and matlab version. This thesis will choose python version. Firstly, clone Faster-Rcnn and compile the Cython module, at last, compile the caffe and pycaffe.

4.Test py-faster-rcnn. In order to test the py-faster-rcnn if runs rightly, this thesis use PASCAL VOC_2007 that pre-trained by the official to have a test. After download faster_rcnn_final.caffemodel.tgz and unpack it, we get caffemodel and VGG_CNN_M_1024 faster_rcnn_final.

4.2.3 Parameter file configuration

This section introduces some core configuration files involved in the training.

1. Set IMDB sub class

The required files are mainly in the datasets directory, and there are three files, factory.py, imbd.py and pascal_voc. Among them, factory.py is a factory class to generate imdb class and return database to provide network training and testing[31]. Imdb.py is basic class of database reading and writing. It encapsulates a lot of database operations. Pasca_voc.py is mainly training class to read and use training data.

2. Configuration factory.py

The main tasks of this layer are: Adapt various data sets using the factory pattern. Use lambda function in factory.py. A custom class that adapts its own dataset and inherits from imdb. The ROI database is mainly for sentient beings in the data set. For each picture, keep all box coordinates and their categories contained in the picture, and then save its area and other parameters by the way. Finally, record the index of all pictures and the method to get the absolute address according to the index.

3. The times of training iterations setting:

Setting the times of training iteration in train_faster_rcnn_alt_opt.py setting the parameter is max_iters = [80000, 40000, 80000, 40000]. Corresponding to the first stage of RPN, the first stage of faster-RCNN, first stage of RPN and the first stage of faster RCNN respectively.

function	Function description
def_laod_image_set_index(self)	Load list file
get gt_roidb(self)	Read and return ground_turth
def selective_search_roidb	Read and return database of ROI, mainly be used in faster-rcnn training
def _load_selective_search_roidb(self,gt_roidb)	Load bounding box files
def init(self,image_set,year,devikit_path=None)	Initial the function
def image_path_at(self,i)	Call image_path_from_index(self, index)
def image_path_from_index (self, index)	Implement the function image_path
def selective_search_IJCV_roidb(self)	Read database of Ground_truth and ROI
def_load_pascal_annotation (self, index)	Read and build gt
def_write_voc_results_file(self,all_boxes)	Write the detections result to the file

TABLE 1. THE LIST OF FUNCTION IN PASCA_VOC

4.2.4 RPN model training

The RPN network training is the first step of faster- RCNN training. The main idea is to initialize the RPN network with the model and then train it. The main function used in training is Train_rpn function.

Parameter setting, table 2

Function	attribute
Cfg.TRAIN.HAS_RPN	TURE
Cfg.TRAIN. BBOX_REG	False
Cfg.TRAIN. PROPOSAL_METHOD	‘gt’
Cfg.TRAIN.IMS_PER_BATCH	1

TABLE 2 THE CONFIGURATION TABLE OF TRAIN_RPN PARAMETER

The important thing here is to set the cfg-train. PROPOSAL_METHOD parameter to 'gt', and the reasons for this are explained below. After setting the basic parameters, the next step is to obtain the training data in imdb and roidb formats.

Let's first introduce what imdb and roidb are. Imdb is a picture database class, containing the name of the database; Roidb is the ROI database, which is actually the target detection bounding box.

Class	Description
Boxes	A two- dimensional array, each row storing xmin, ymin, xmax, ymax, the row refers to the number of multiple boxes
Gt_classes	Include box index
overlap	A two-dimensional array of row number refers to the box, there were a total of 21 column, storage is 0.0 or 1.0, when the box corresponding category, natural 1.0 this actually means for ground way box, so after natural overlap is 1, and other natural overlap is 1, after compared with other natural overlap to 0, was later turned sparse matrix
seg_areas	Save the areas of bounding box
flipped: false	Show that the images are not flipped

TABLE 3 THE LIST OF MEMBERS OF CLASS ROIDB

Getting training data mainly uses the get_roidb () function which has been used for the roidb data object. First, it finds the cache with the 'PKL' extension in the cache path, which serializes the roidb through the cpickle tool. If the file exists, it will read the contents here first for efficiency. Otherwise it will call this private function called _load_pascal_annotationc to load the data in the roidb, save it in the cache file, and return the roidb.

The important training data needed in get_roidb () function is imdb. However, pascal_voc () function is called in get_imdb () function to create imdb data. It mainly

sets the path of data set, index of picture name and so on, but does not store the actual picture information. In fact, the psacal_voc class is a subclass of the imdb class; When the imdb data is obtained, the get_roidb () function immediately requests a method to set the proposed area to the set_proposal_method () function, also thinking of adding roidb data to the imdb, which uses the function set_proposal_method().

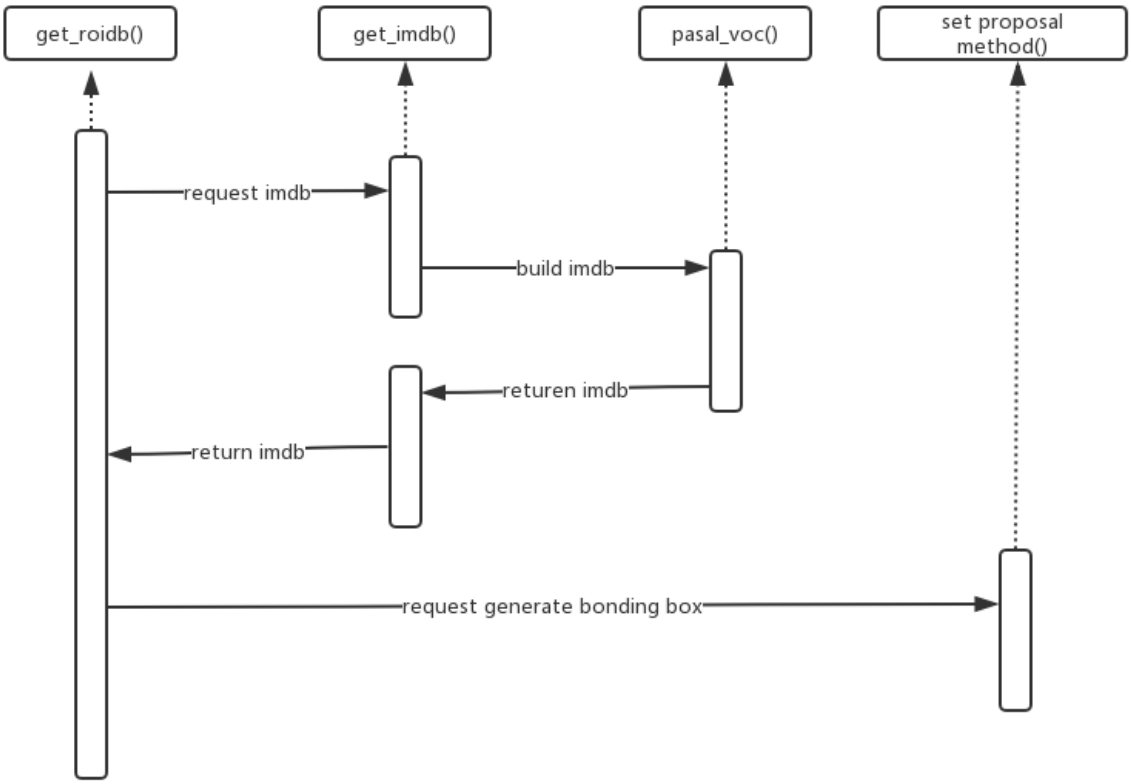


FIG.12. THE SEQUENCE DIAGRAM OF DATA REQUEST

In a function set_proposal_method (), the whole process is to parse the data in eval () that makes it valid, and then pass it on to roidb_handler. In the figure 13, firstly, we use function train_rpn(), because it sets cfg. TRAIN.PROPOSAL_METHOD ='gt', it's a method to parse the data. That's why we set parameter like that.

Next, we set cfg. TRAIN.PROPOSAL_METHOD parameter to request gt_roidb () though train_rpn: this thesis use function _load_pascal_annotation () to get roi of ground truth through parse XML file. In _loadpascal_annotationfunction, according to the index of each image, to the Annotations of this folder to find the corresponding XML tagging data, and then to load all the bounding box of object, and remove all the complex object. At this time, the original roidb format data was obtained from imdb, but this is not the roidb data in training.

After getting the data in roidb format, the next step is to get the final training data；

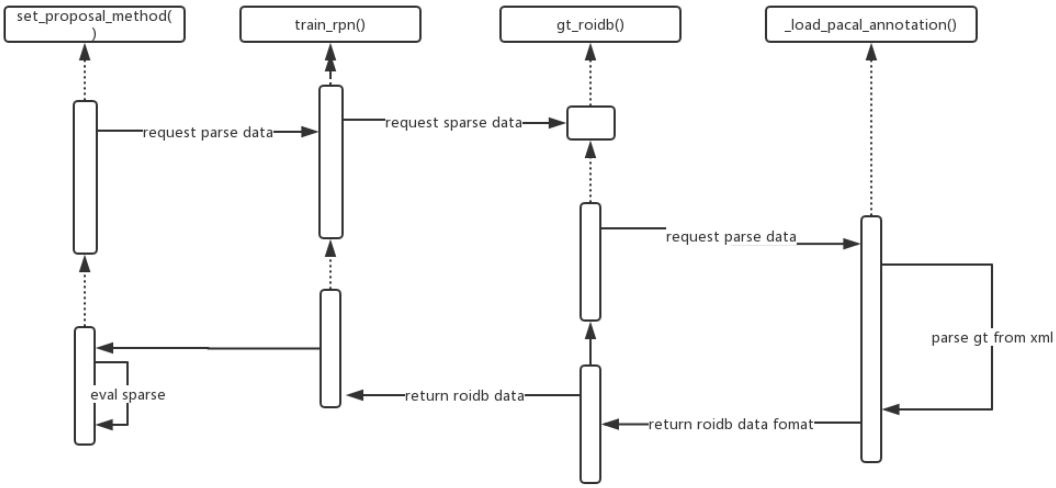


FIG.13. THE PROCEDURE OF ROIDB PARSING

As shown in figure 14, after getting the data in the original roidb format, it will continue to the get_roidb () function, and the roidb data finally used for training will be obtained through the get_training_roidb () function.

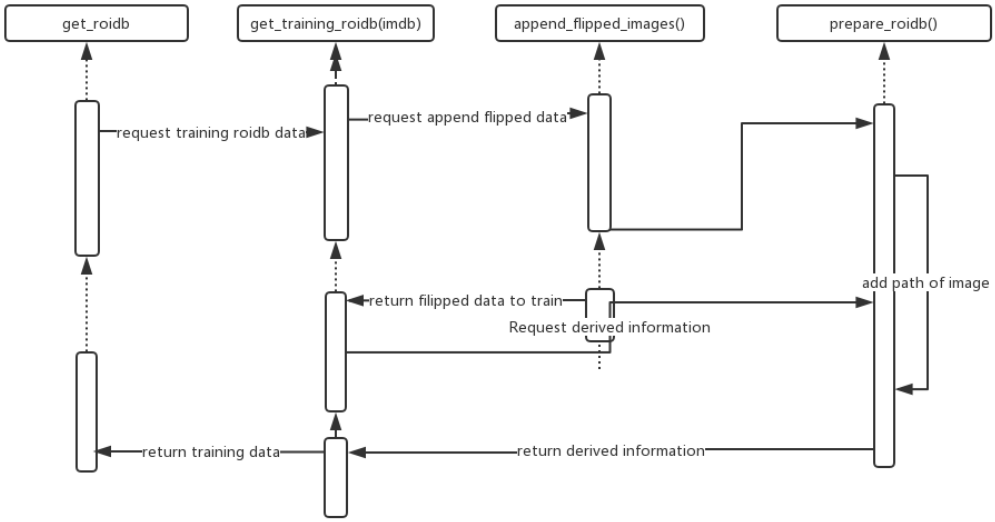


FIG.14. THE SEQUENCE DIAGRAM OF TRAINING DATA ACQUISITION

4.2.5 Fast-RCNN training

The previous section mainly introduced the training process of RPN in a series of details, and the following section mainly introduced the training process of the faster-rcnn network and its related technical details.

The overall training process uses the RPN that has been trained in the previous step to generate proposals, and then inputs the proposals generated by RPN into the network for training.

First, how to use the RPN network that has been trained in the first step to generate anchors. In the figure 15, the proposal of generation mainly used function rpn_generate (), the process:

1. First, set up the pre-NMS, which will generate about 2000 proposals after passing through the NMS.

2. After obtaining these proposals, initialize caffe and then use the get_imdb () function to get the imdb data.

3. The whole RPN network is loaded with the caffe_NET () method, using imdb_proposals () are made in the region

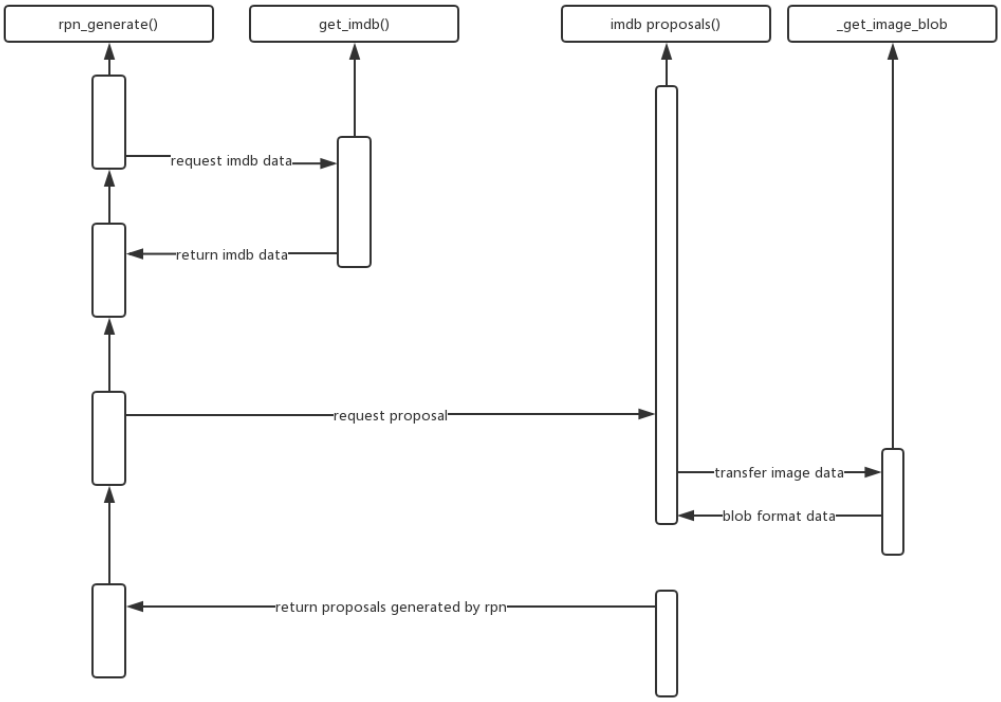


FIG.15. PROPOSAL GENERATION SEQUENCE DIAGRAM

4. The next step is to use the generated proposal to train Fast-RCNN network. The main function is train_fast_rcnn ().

Mainly using function rpn_roidb, parameter setting like that:

function	parameter
Cfg.TRAIN.HAS_RPN	False
Cfg.TRAIN. PROPOSAL_METHOD	rpn
Cfg.TRAIN.IMS_PER_BATCH	2

TABLE 4. THE CONFIGURATION OF RPN_ROIDB PARAMETER

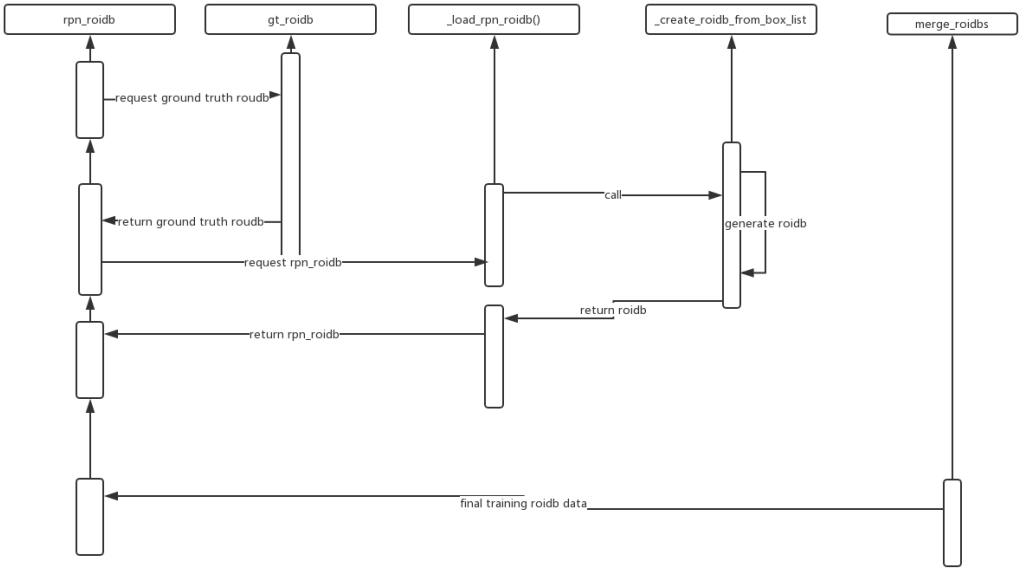


FIG. 16. THE SEQUENCE DIAGRAM OF DATA PREPARATION

As shown in figure 16, this method first obtains the roidb with ground truth through the gt_roidb method, and then obtains the roidb generated by rpn-using _load_rpn_roidb () (in the _load_rpn_roudb () method, by mobilizing the create_roidb_from_box_list () function to generate the roidb data). Where box_list is an array, each element is a list, and each list refers to the boxes contained in an image. The method also defines:

1. Bbox_overlaps: Each box of a proposal does a coincidence degree calculation with the box of ground-truth.

2. Overlap = (overlap area)/ (proposal_box area +gt_boxes area - overlapping part area) for each proposal, the corresponding category of the largest gt_boxes is selected, and then the corresponding overlapping value is filled in under the corresponding class index.

After the rpn-generated proposal was made into roidb, it was then returned to rpn_roidb (), and merge_roidbs () was used to combine the previously obtained gt_roidb and rpn_db, and finally the roidb data required by fast RCNN for final training was obtained.

After the data needed for training are ready, we return to train_fast_rcnn () for training, mainly using the train_net function.

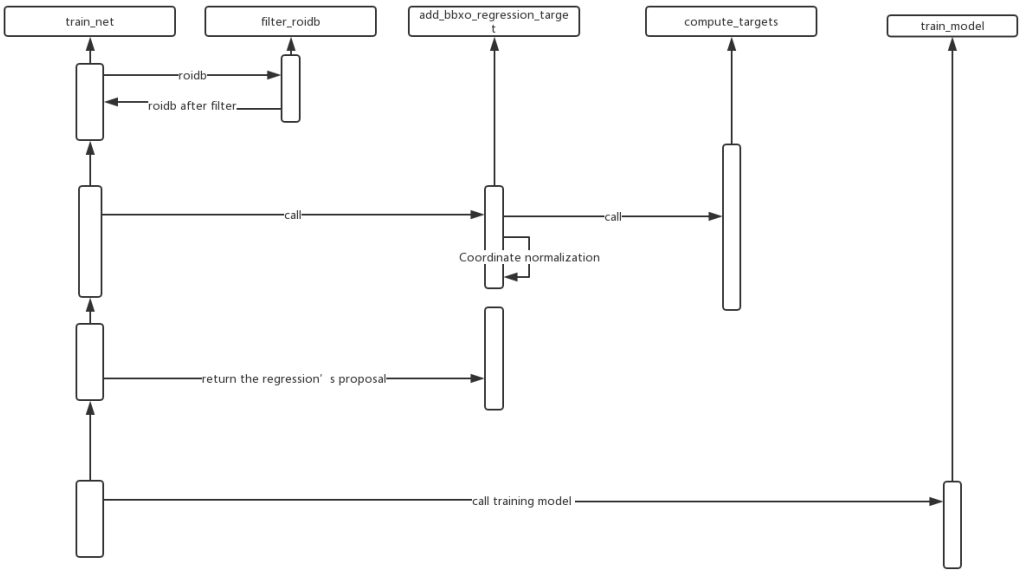


FIG. 17. THE SEQUENCE DIAGRAM OF FAST-RCNN TRAINING

As shown in figure 17, the first is to use filter_roidb before () to produce again for training fast_rcnn roidb screening, screening, after back to train_net () function, create a solverWrapper object, which is training the network model, in this class, there is a add_bbox_regression_targets () function, the function of it is produced for the RPN proposal to provide return properties, This function adds another key to roidb: 'bbox_targets'. The main function of this function is the _computer_targets () function, which is mainly used to generate regression attributes. Other parts are mainly used to

get the coordinates of rpn_roidb and the mean and variance, which can be used for coordinate normalization.

Finally, the train_model () method is invoked to obtain the Fast-RCNN network training model.

At this point, the model training process of RPN and Fast-RCNN network is over. Later, the same method is adopted to continue to use the Fast-RCNN training model to initialize RPN, but the conv layer parameters are kept unchanged, and then the proposal is generated based on the RPN network. Finally, Fast-RCNN is trained and the final model is output.

4.3 Training fine tune

4.3.1 Data preprocessing

Fast RCNN training mainly needs three kinds of data: image data, annotated data (i.e., Ground Truth) and image bounding boxes that are pre-processed by RPN.

First, use the python script to traverse the image path in the sample folder and store it in the TXT file. Then traverse the annotated TXT file stored in a unified folder, read out the Ground Truth information of each picture with a regular expression, and write annotation.txt file in the following format:

[image path] [target number] [target 1 bounding box] [target 2 bounding box] …

As shown in the figure 18:

```
lfw_5590/Aaron_Eckhart_0001.jpg 84 161 92 169 106.250000 107.750000 146.750000 112.250000 125.250000 142.750000 105.250000 157.750000 139.750000
161.750000
lfw_5590/Aaron_Guiel_0001.jpg 85 172 93 181 100.250000 111.250000 145.750000 116.750000 124.250000 136.750000 92.750000 159.750000 138.750000
163.750000
lfw_5590/Aaron_Peirsol_0001.jpg 88 173 94 179 106.750000 113.250000 146.750000 113.250000 129.250000 139.750000 108.250000 153.250000 146.750000
152.750000
lfw_5590/Aaron_Pena_0001.jpg 67 176 83 192 101.750000 116.750000 145.250000 103.750000 125.250000 136.750000 119.750000 163.750000 146.250000
155.750000
lfw_5590/Aaron_Sorkin_0001.jpg 73 164 86 178 101.250000 112.250000 143.750000 111.750000 129.750000 137.750000 99.250000 155.250000 142.250000
154.750000
lfw_5590/Aaron_Tippin_0001.jpg 77 165 90 177 103.750000 111.750000 141.750000 111.250000 129.250000 133.750000 110.750000 156.250000 134.750000
155.250000
lfw_5590/Abba_Eban_0001.jpg 80 164 90 175 105.750000 113.750000 145.750000 110.250000 133.750000 136.250000 107.750000 156.250000 148.750000 155.250000
lfw_5590/Abdel_Aziz_Al-Hakim_0001.jpg 86 169 90 173 110.750000 110.250000 148.250000 113.750000 136.750000 133.250000 111.750000 152.250000 141.750000
154.250000
lfw_5590/Abdel_Nasser_Assidi_0001.jpg 80 167 87 174 103.750000 112.250000 148.250000 115.250000 124.250000 134.250000 106.750000 155.750000 136.250000
159.250000
```

FIG18. DATA ANNOTATION FORMAT

4.3.2 Deep neural network pre-training and fine-tuning

The algorithm in this paper adopts Faster-RCNN, and the training process is carried out on the server of Ubuntu 14.04 LTS with the deep learning framework of Caffe.

CaffeNet and VGG_CNN_M_1024 based on the network structure adopted in this paper are tested on PASCAL_VOC target detection data set, and pre-trained on ImageNet general target detection data set.

(1). The CaffeNet used in this paper is a reconstruction of Caffe framework's landmark AlexNet proposed by Alex in the ImageNet image classification challenge in 2012. Its structure is shown in the figure 19, with 5 convolution layers and 3 full connection layers:

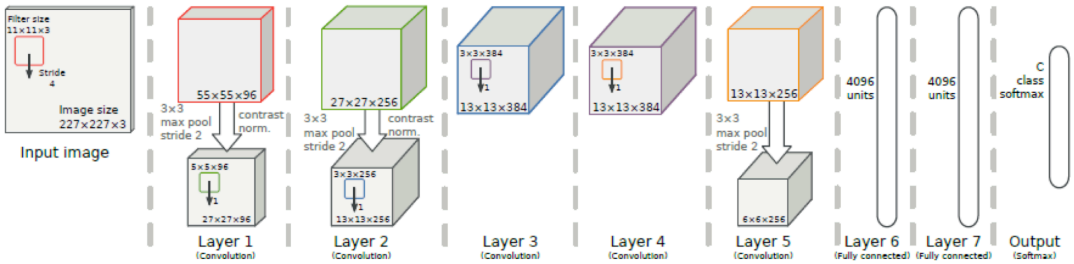


FIG.19. CAFFENET NETWORK STRUCTURE DIAGRAM[31]

CNN-M	96x7x7 st. 2, pad 0 LRN, x2 pool	256x5x5 st. 2, pad 1 LRN, x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max

FIG20. VGG_CNN_M1024 NETWORK STRUCTURE DIAGRAM[32]

The VGG_CNN_M_1024 depth network structure was proposed by Ken Chatfield et al.[32], which is smaller than the famous VGG16 depth network, and the structure is shown in figure 20.

It can be seen that this network is similar to ZF network, with 5 convolution layers and 3 full connection layers. The setting of the full connection layer of the two networks is very similar, the main difference is that the number and size of convolution kernel are different in the convolution layer, and the convolution layer of VGG_CNN_M_1024 network is "wider".

(2). Since there are 20 target categories of PASCAL_VOC, and only one pedestrian needs to be detected in this paper, the network structure is firstly adjusted accordingly. On the basis of the original network, the category of classification is changed from "20 categories plus background" to "pedestrian plus background", that is, by

CLASSES = ('__background__', 'aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor')

Transfer it to:

CLASSES = ('__background__','person')

The corresponding structure of the network was adjusted. Firstly, num_classes in the input data layer was changed from 21 classes to 2 classes. Then, num_output of cls_score layer, which outputs the score of each class, is changed from 21 to 2. Finally, the num_output number of bbox_pred layer that outputs the ROI position of each category is changed from 84 to 8 (because each bounding box has 4 values of [xmin ymin xmax ymax]). The files of factory.py, pascal_voc. py and succise.py under fast-rcnn/lib/datasets/ folder are modified correspondingly (see appendix for details). A new Caltech class containing custom methods is created in python, thus completing the corresponding changes of network structure and interface.

(3). Then the deep network pre-training. In this paper, the training of small network CaffeNet and medium network VGG_CNN_M_1024 (at least 3G memory) is limited by the computing capacity of the server, and the training of VGG16 network (at least 11G memory) is not available[33]. During the training, the initial value of network weights loaded the network weights that the network trained 40,000 times on the general target dataset ImageNet, that is to say, the two networks pre-trained on the general target detection dataset ImageNet.

```
train_net: “models/CaffeNet/train.prototxt”
base_lr: 0.001
lr_policy: “step”
Gamma: 0.1
Stepsize: 30000
Diaplay : 20
Average _loss: 100
Momentum: 0.9
weight_decay: 0.0005
#we disable standard caffe solver snapshotting and implement our snapshotting
#function
Snapshot : 0
#We still use the snapshot prefix, though
 snapshot_prefix: “caffenet_fast_rcnn_INRIA”
#debug_into: true
```

FIG.21. NETWORK TRAINING RELATED PARAMETERS

(4). The next official start deep network training(fine-tuning). In fact, the network training in this paper was fine-tuning of the tasks of human inspection of deep network needles on the basis of ImageNet pre-training. In the training, 614 pedestrian images (2416 pedestrians in total) were concentrated in INRIA training, while 288 pedestrian images (1126 pedestrians in total) were concentrated in INRIA testing. Relevant parameters during training are shown in the figure 21.

Among them, the basic learning rate is 0.001 and the maximum number of iterations is 40,000. For training, CaffeNet takes about 0.17 seconds per iteration, and VGG_CNN_M_1024 takes about 0.26 seconds.

5. System test and analysis

5.1 The test method

The performance criteria for pedestrian detection algorithms are the same as those for target detection algorithms, which need to consider three aspects: detection quality, time complexity and space complexity. The time complexity is mainly reflected in the algorithm's operation speed, while the space complexity is mainly reflected in the storage space (including RAM and video memory) that the algorithm needs to occupy, while the detection quality is mainly reflected in whether the pedestrian in the input image can be successfully detected and whether the obtained location is accurate.

5.1.1 Time complexity and space complexity

The time complexity and space complexity of Faster-RCNN are obviously closely related to the number of convolution layers, the number and size of convolution kernel in each layer, the number of full connection layers and the number of neurons in each layer [33]. Once the network structure is determined, the storage space required by the algorithm is determined. The execution time of the algorithm is also related to some other factors, such as whether to use GPU to accelerate parallel computation (the speed can be increased by tens of times than the CPU) and the number of candidate region proposals generated by selective search algorithm, etc. The two criteria are also clear: the amount of storage (KB) and the average computing time per image (ms).

5.1.2 Detection quality

For a detector, the measurement of detection quality is more complicated than the calculation time and storage space. In the case of pedestrian detection, the detection algorithm performs binary classification on the target window, that is, whether the window contains pedestrians or not. Therefore, there may be four situations in the test:

TP -- judge pedestrians as pedestrians;

FN -- judge pedestrians as non-pedestrians;

FP -- judge non-pedestrians as pedestrians;

TN -- judge delinquent as delinquent.

In the classifier, the Miss Rate (also known as the false detection Rate), an important indicator used to measure pedestrian detection, is defined as the ratio of the number of pedestrians judged as non-pedestrians to the number of head office:

$$MR = FN/(TP + FN) \tag{21}$$

However, the loss rate alone is not enough to reflect the performance of the algorithm, because non-pedestrians are judged to be pedestrians. The extreme case that can be imagined is that an algorithm judges all inputs, whether pedestrians or not, as pedestrians, and then MR is 0, but obviously the performance of this algorithm is very poor. Another indicator to consider is the relationship between the frequency of FP and MR. If an algorithm judges whether the window contains pedestrians at a lower threshold, it is more inclined to judge the window as containing pedestrians, then the frequency of FP (non-pedestrians are judged as pedestrians) is higher, while MR is lower. Therefore, the frequency of FP is negatively correlated with that of MR. For an algorithm, when the frequency of FP is a constant value, the lower MR is, the better the algorithm performance will be. Therefore, the evaluation standard used to measure the performance of pedestrian detection algorithm is the relationship between FP frequency and MR. As shown in the figure 22, the horizontal axis is FPPI (False Positive Per Image, FP in each Image), and the vertical axis is MR. With the increase of FPPI, the threshold of classifier decreases, which leads to the decrease of the loss rate.

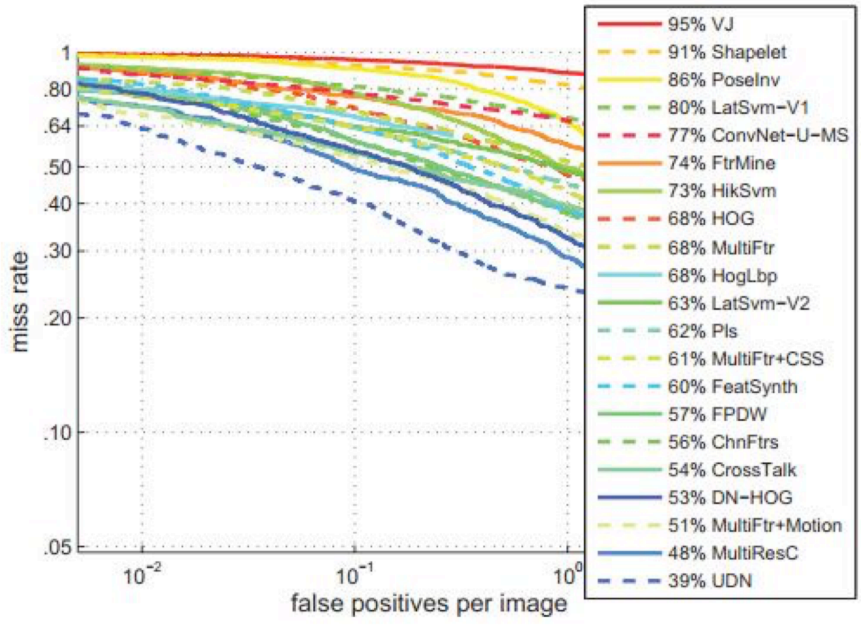


FIG22. PEDESTRIAN DETECTION ALGORITHM PERFORMANCE INDEX, MR AND FPPI RELATIONSHIP CURVE[34]

Similarly, similar indicators are required to measure the performance of the algorithm in the detector. Generally, it is measured by Precision and Recall, which are defined as:

$$P = \frac{TP}{TP+FP} \tag{22}$$

$$R = \frac{TP}{TP+FN} = 1 - MR \tag{23}$$

The meaning of accuracy is the proportion of pedestrians in the window detected by the detector, while the meaning of recall rate is the proportion of all pedestrians detected. The ideal detector should have a high accuracy and recall rate, but the actual situation may not be. Similar to the relationship between MR and FPPI, if the threshold of the detector is higher, the probability that the detected window is pedestrian is higher, that is, P increases, but some Windows with lower scores including pedestrian will not be detected, leading to a decrease in R. So, there's also a negative correlation between P and R.

Another difference between the detector and the classifier is that the detector also needs to give the location information of pedestrians, and the location information obtained by the detector is often not completely consistent with the actual Ground Truth. So, when a window frames the pedestrians but does not completely coincide with the Ground Truth, how to judge whether it can be counted as correct? The concept of IoU (banding -over-Union) was used in the paper proposing RCNN:

$$IoU = Area/(Areal1 + Area2 - Area) \tag{24}$$

Area1 and Area2 are the Area of two Windows respectively, and Area is the Area of their intersection, so IoU is the ratio of the intersection Area and union Area of two Windows. Therefore, it can be defined that when the IoU of the Proposal of a window and the Ground Truth of the actual position are greater than a certain threshold, the window is considered to be correct.

5.1.3 Test method

Just like the training process, the Faster-RCNN algorithm performance was tested by using 287 images of 1148 pedestrians in the INRIA test data set on the server with Ubuntu 14.04 LTS and NVIDIA 1070 GPU (memory capacity 8096 MB):

(1). Test the storage space required for algorithm execution;

(2). Test the operation time of the algorithm for each image in the running process;

(3). The accuracy P and recall R of the detector under different IoU standards were tested.

The above contents have been tested on CaffeNet and VGG_CNN_M_1024 network respectively, and the results and analysis are in the next section.

5.2 Test and analysis

5.2.1 Run time and required storage space

	CaffeNet	VGG_CNN_M_1024
Average test time(s)	0.155832	0.228933
Storage space(kb)	7335472	19276464
Document generation training time(s)	0.10	0.15

TABLE5. RUN TIME AND REQUIRED STORAGE SPACE

It can be seen that on the server, the processing time of each picture is about 0.2s, which is different from the literature running time to some extent. This is mainly caused by the different models of GPU.

5.2.2 Detection quality

In order to compare the performance of different network differences and tuning (fine-tuning) before and after the process of network performance difference. In this paper, the different before and after the IoU, different network structure and network tuning accuracy under the condition of precision and called back rate recall were tested, respectively, and also to the traditional detection algorithm HOG + SVM under the same conditions in the test as comparison. Testing quality of CaffeNet and VGG_CNN_M_1024 network after 40,000 tuning iterations is shown in table 7.

At the same time, the deep network structure of CaffeNet and VGG_CNN_M_1024 (referred to as VGGNet in the table) was also tested before tuning, that is, the detection quality of the network that has been pre-trained only on the general target detection data set of ImageNet, and the detection quality of the training iterations of 10000, 20000 and 30000 times, as shown in table 6, table 7, and table 8.

The precision and recall rates of the two kinds of deep network structures after tuning under different IoU thresholds.

IoU threshold	Precision			Recall		
	CaffeNet	VGGNet	HOG+SVM	CaffeNet	VGGNET	HOG+SVM
0.1	0.9521	0.9671	0.7853	0.8452	0.7993	0.8895
0.3	0.9502	0.9502	0.6456	0.8435	0.7976	0.7312
0.5	0.9349	0.9568	0.2072	0.8299	0.7908	0.2347
0.7	0.7854	0.8045	0.0541	0.6973	0.6650	0.0612
0.8	0.5249	0.5597	0.0390	0.4669	0.4626	0.0442
Ave.	0.8295	0.8506	0.3462	0.7364	0.70321	0.3922

TABLE 6 CAFFENET DEEP NETWORK TUNING EACH STAGE IN DIFFERENT IOU THRESHOLD ACCURACY AND RECALL RATE

IoU threshold	Precision			Recall		
	10000	20000	30000	10000	20000	30000
0.1	0.9401	0.9591	0.9540	0.8537	0.8384	0.8469
0.3	0.9401	0.9572	0.9521	0.8537	0.8367	0.8452
0.5	0.9213	0.9358	0.9406	0.8367	0.8180	0.8350
0.7	0.7854	0.8045	0.0541	0.6973	0.6769	0.6922
0.8	0.5249	0.5597	0.0390	0.4524	0.4490	0.4439
Ave.	0.8120	0.8280	0.8253	0.7374	0.7238	0.7326

TABLE 7 VGGNET DEEP NETWORK TUNING EACH STAGE IN DIFFERENT IOU THRESHOLD ACCURACY AND RECALL RATE

IoU threshold	Precision			Recall		
	10000	20000	30000	10000	20000	30000
0.1	0.9529	0.9695	0.9547	0.8265	0.8384	0.8469
0.3	0.9401	0.9695	0.9528	0.8214	0.8112	0.8231
0.5	0.9213	0.9595	0.9409	0.8146	0.8027	0.8129
0.7	0.7804	0.8110	0.7953	0.6769	0.6786	0.6871
0.8	0.5249	0.5591	0.5689	0.4592	0.4762	0.4915
Ave.	0.8298	0.8557	0.8425	0.7197	0.7160	0.7279

TABELE.8 TWO KINDS OF DEEP NETWORK STRUCTURE ARE USED TO OPTIMIZE THE ACCURACY OF EACH STAGE UNDER DIFFERENT IOU

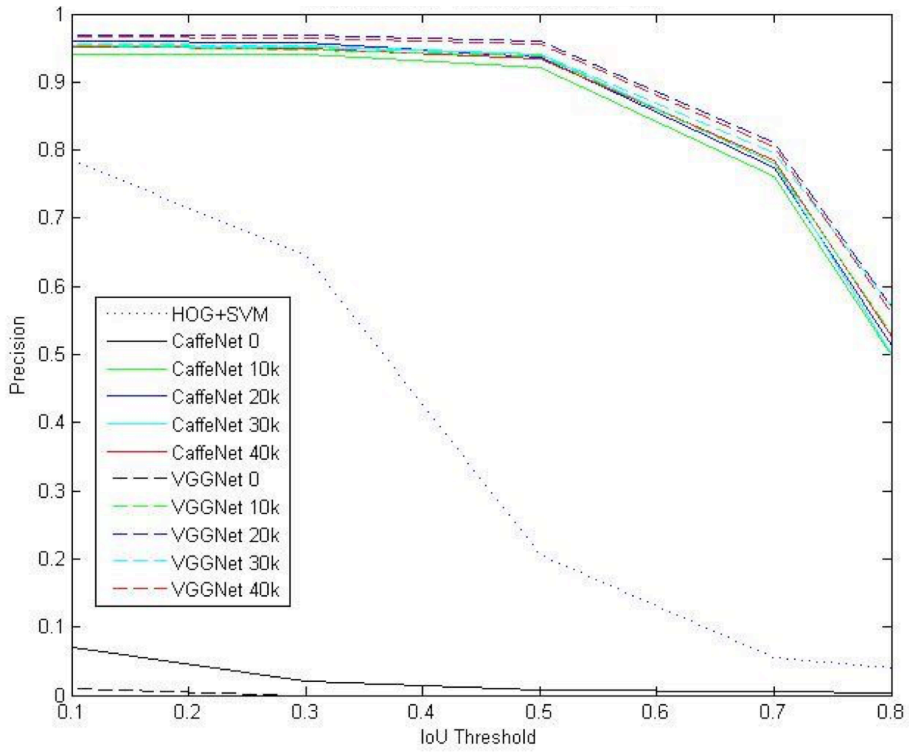


FIG.23 TWO KINDS OF DEEP NETWORK STRUCTURE ARE USED TO OPTIMIZE THE PRECIOUS OF EACH STAGE UNDER DIFFERENT IOU

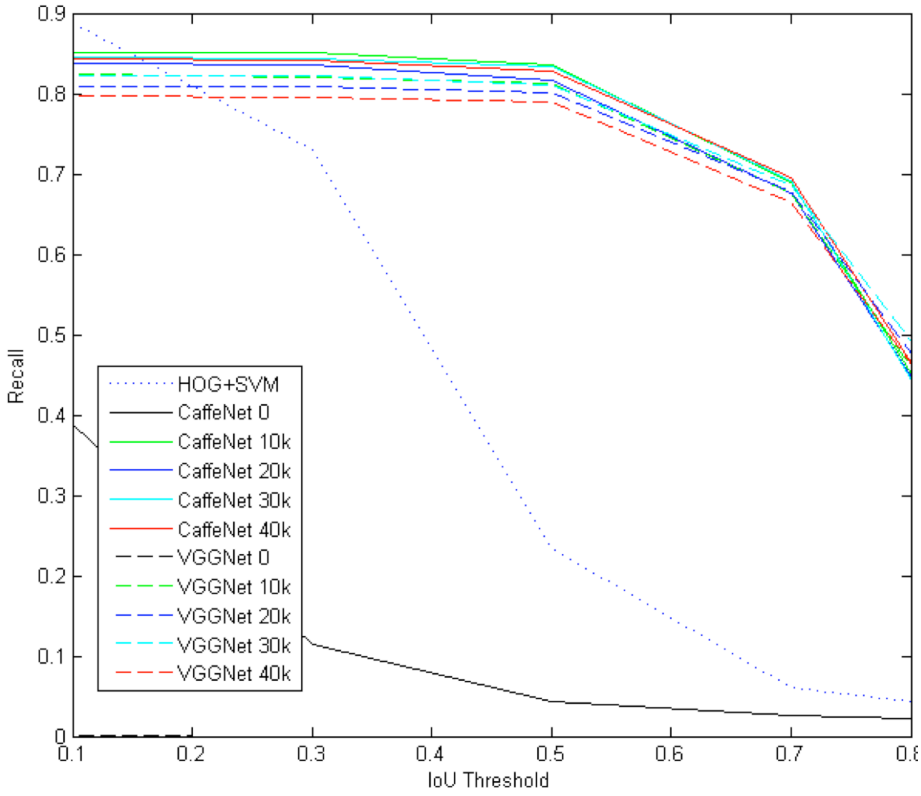


FIG.24 TWO KINDS OF DEEP NETWORK STRUCTURE ARE USED TO OPTIMIZE THE RECALL OF EACH STAGE UNDER DIFFERENT IOU

The curves of the precision of CaffeNet and VGG_CNN_M_1024 network with the threshold value of IoU are shown in figure 23 and figure 24 respectively.

The following conclusions can be drawn from the table data and charts:

(1). From the perspective of different algorithms, by comparing the performance of CaffeNet and VGGNet for depth network detection algorithm with the HOG+SVM of the widely used traditional detection algorithm, it can be found that the representation of depth network algorithm is far beyond HOG+SVM. The average precision of the deep network algorithm is more than 80%, while the AP of HOG+SVM is less than 35%. The average recall rate of deep network is more than 70%, while the average recall rate of HOG+SVM is less than 40%. From two aspects of indicators, the performance of pedestrian detection algorithm based on deep learning is indeed far beyond the traditional algorithm.

(2). In terms of the structure of deep network, after network tuning, the average accuracy of VGG_CNN_M_1024 network and CaffeNet network at IoU values of 0.1, 0.3, 0.5, 0.7 and 0.8 are respectively 85.06% and 82.95%. The AP of VGG_CNN_M_1024 is about 2% higher than that of CaffeNet network. The average recall rate at the above 5 IoU thresholds was 70.31% and 73.64%, respectively. The recall rate of VGG network was about 3% lower than CaffeNet's. Therefore, in general, CaffeNet tends to produce more proposals with relatively low quality, and cover more Ground Truth by making up for the lack of quality through quantity. However, VGG_CNN_M_1024 network tends to produce fewer but better proposals, so it has lower recall rate and higher precision[35].

(3). from the perspective of pre-training and tuning, tuning can significantly improve network performance. As shown in table 3, table 4 and figure 25, before tuning, only for training network detection quality is very poor, CaffeNet accuracy is 2.13%, the recall rate was 11.9%, VGGNet accuracy rate and recall rate less than 1%, even close to completely without the level of the trained network, general target detector for in particular the completion of tasks of pedestrian detection capability is not strong. However, there is a great potential for network performance improvement. After only 10,000 iterations, the network performance has basically reached a very high level, indicating that these networks need only a little training to achieve a great improvement in performance.

(4). in the paper that Ross proposed Fast-RCNN algorithm, Fast-RCNN was trained and tested on PASCAL_VOC(Visual Object Classes Challenge)2007, 2010 and

2012 data sets (including 20 categories of objects), and VGG16 network (larger than VGG_CNN_M_1024) was used as Fast RCNN's deep network[34]. Accuracy in three data sets on macro level (mean Average Precision) are 70.0%, 68.8% and 68.4% respectively, and specific to the line on this category, detection accuracy (AP) were 69.9%, 72.7% and 72.0%, respectively, so compared with the data, this article training network both CaffeNet VGG_CNN_M_1024 gained by the depth, the Average accuracy is over 82%, network performance has been increased greatly.

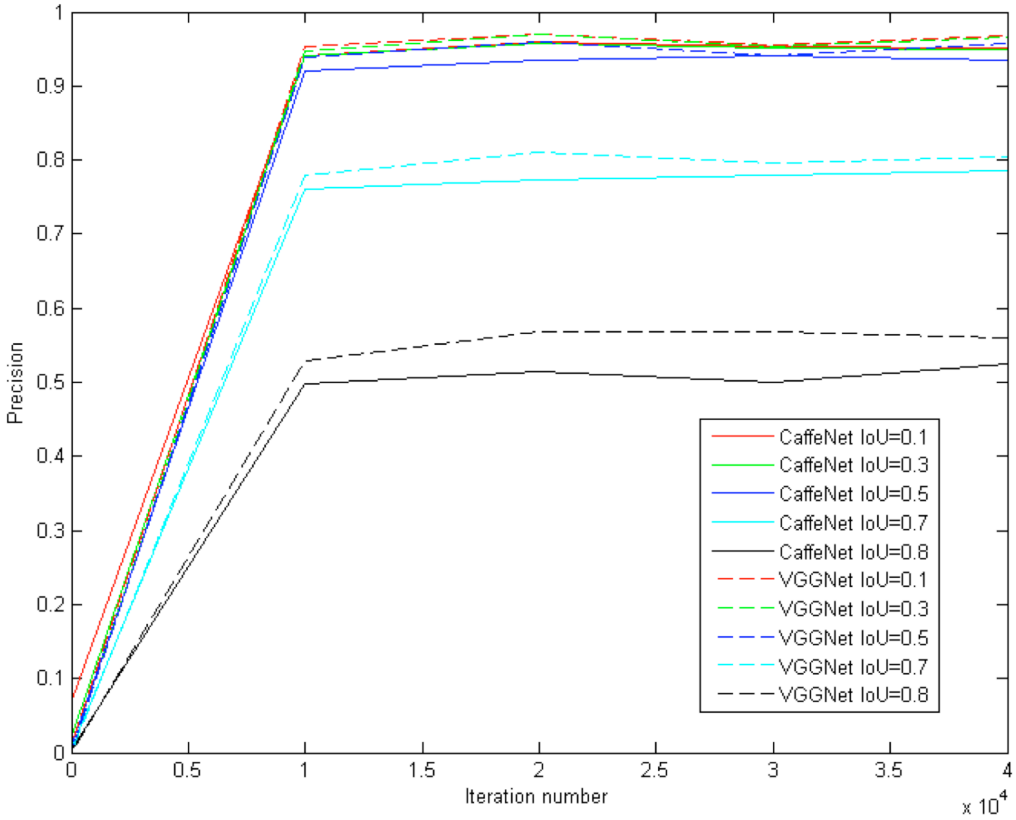


FIG. 25 VARIATION CURVE OF PEDESTRIAN DETECTION ACCURACY WITH THE NUMBER OF TRAINING ITERATIONS

(5). Can be seen from the figure 25, after training 10000 times iteration, the performance of the network has been basically reached the stable level, and then tens of thousands of times of training, to improve network performance are very limited, even from 20000 to 30000, the performance of the network and a slight decline, suggesting that the training of the network appeared over fitting phenomenon (namely network on the training set the fitting degree of progress, but a worse and worse performance on the test set). The main reason for this phenomenon should be that the data set adopted in the training is not large enough. As the training set adopted only

contains 614 pedestrian images (a total of 2416 pedestrians), the limited number of samples leads to the fact that the network's training on the training set cannot further improve its performance. Therefore, in the follow-up work, pedestrian data sets with larger data volume will be used to train the network, so as to obtain better detection results.

5.3 Training output analysis

Completed based on the depth of the neural network training and testing process of detector, use the INRIA pedestrians in a Caffe environment image data set is completed CaffeNet and VGG_CNN_M_1024 two kinds of network structure of the training, the detection performance of the detector is obtained by the test index, published its than 2015 Fast RCNN algorithms have the bigger progress, an average of about 10% accuracy[35].

The following TEST images are from the test-set of the INRIA pedestrian image data set. The red rectangle box is the pedestrian location information (namely Ground Truth) provided by the database itself, and the green rectangle box is the pedestrian location calculated by Fast RCNN detector in this paper.

It can be seen from figure 26, 27 and 28 that the algorithm can successfully capture pedestrians in the picture and has a good adaptability to the size of human body in the picture. In FIG. 26, there is a big difference in size between adults and children. In figure 27, several figures have large sizes, and the algorithm can complete the detection well. It can be seen from FIG. 28 and 31that the algorithm performs well in detecting images with only a single pedestrian.

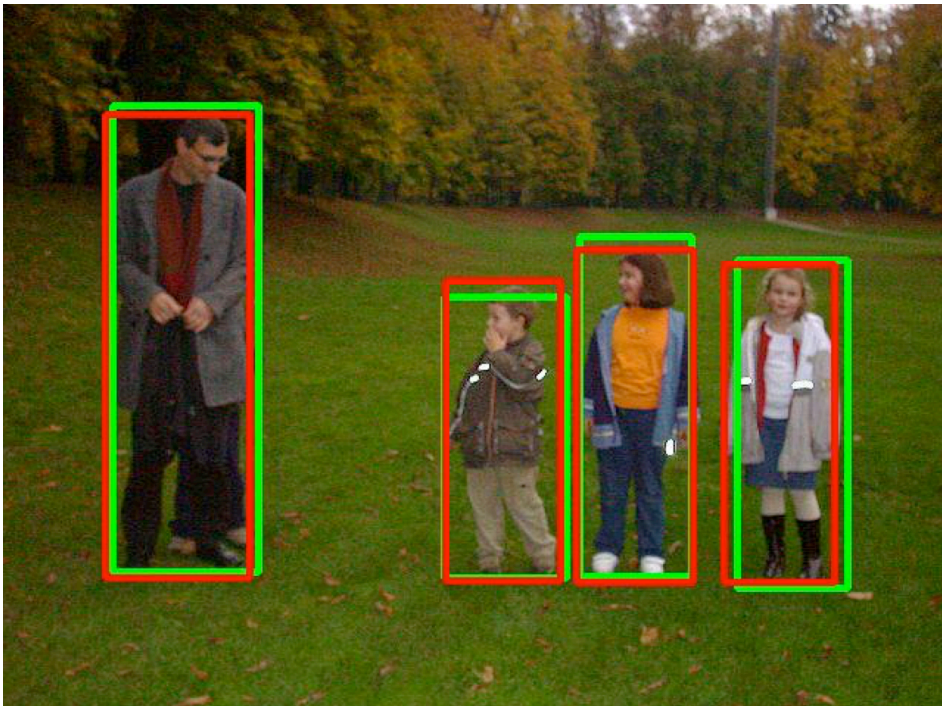


FIG26. PEDESTRIAN DETECTION TEST 1

TEST BOX IS GREEN, GT_BOX IS RED

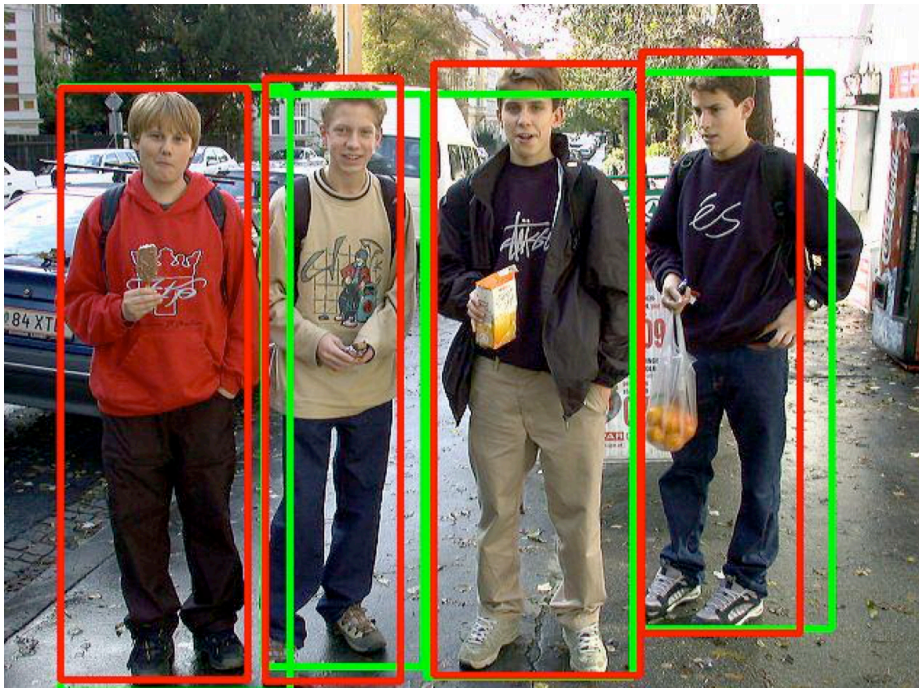


FIG27. PEDESTRIAN DETECTION TEST 2

TEST BOX IS GREEN, GT_BOX IS RED

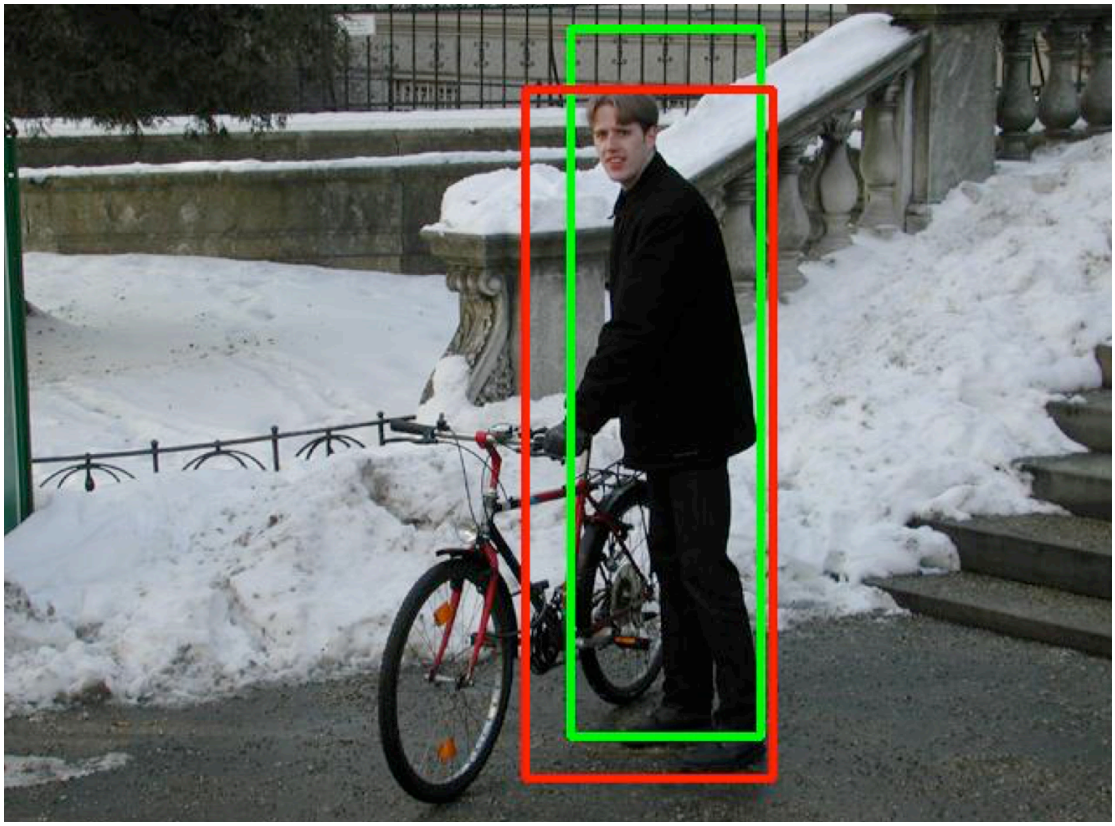


FIG28. PEDESTRIAN DETECTION TEST 3

TEST BOX IS GREEN, GT_BOX IS RED

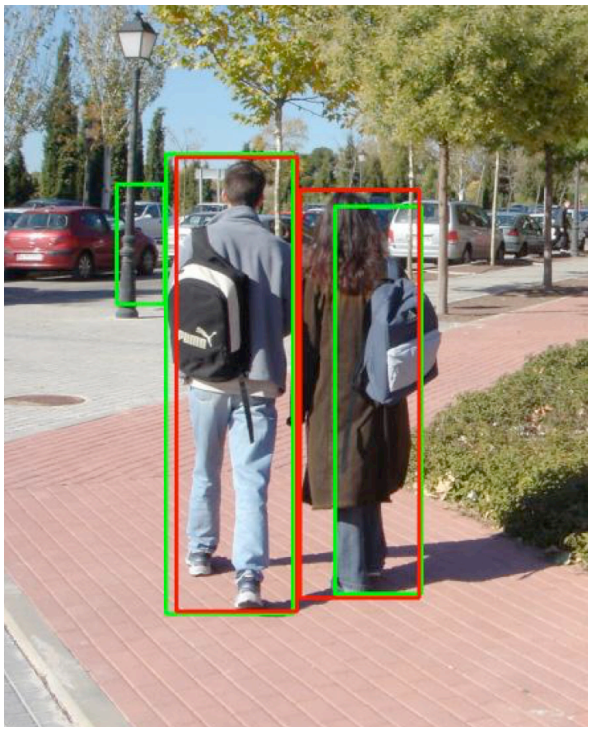


FIG29. PEDESTRIAN DETECTION TEST 3

TEST BOX IS GREEN, GT_BOX IS RED

FIG30. PEDESTRIAN DETECTION TEST 4

TEST BOX IS GREEN, GT_BOX IS RED

FIG31. PEDESTRIAN DETECTION TEST 5

TEST BOX IS GREEN, GT_BOX IS RED

Some of the problems with the algorithm are shown in figures 29 and 30. In figure 29, miss detection occurs, and only three of the four figures are successfully detected. This paper[35] believes that the main reason is that the pedestrians who are missed are blocked by railings, which cannot be classified into the same area when the search algorithm selects the proposal. In figure 30, a false detection occurs, and the wrong area is also detected as pedestrians, which is believed in this paper to be due to the interference caused by the telegraph poles included in the proposal. In figure 30, there was an omission. A figure detected here did not appear in the Ground Truth, which should be an omission by the data set producer.

Though, like in the figure, detection system can appear some residual and the fault detection problems, led to the accuracy and recall rate lower, but overall, on the basis of IoU is greater than 0.5, the accuracy in detection system have reached more than 90%, the recall rate is around 80%, the detection of quality has been well before algorithm has greatly improved, can close to the level of practical use.

6. Conclusion

Because pedestrians are the main participants in the road traffic, also direct victims in traffic accidents, in order to effectively protect the safety of the pedestrians, timely warning drivers may be related to vehicle collision adjacent pedestrian. pedestrian detection technology research and development will be the development of the automotive safety auxiliary driving technology provides powerful theory and technical support, and has potential economic value and application prospect.

In this master thesis, I did the research that is from traditional objective detection method to neural network detector. Through comparison, it is known that the Faster-RCNN exceeds the traditional algorithm in both detection speed and accuracy. So I choose Faster-RCNN as the detection algorithm. I prepared two datasets that PASAL VOC_2007 and INRIA. I used PASAL VOC_2007 dataset to pre-train the RPN and Fast-RCNN model, and used INRIA dataset to fine tune the deep neural network model to get better detection precision. After training process, I got the result that the accuracy in detection system have reached more than 90%, the recall rate is around 80% on the basis of IOU is larger than 0.5. The detection of quality has been well before algorithm has greatly improved and close to the level of practical use.

However, there are still some problems. Through a large number of visualization test results, it can be found that Faster-RCNN is easily confused by background information, resulting in false detection, and these background areas are easily discarded when conducting positive and negative sample calibration. To solve this problem, this paper changes the IOU threshold to minimize the influence of complex background on detection results.

The detection of pedestrian detection targets with less available information is always a big problem in pedestrian detection. In this paper, multi-layer feature fusion is used to optimize the detection results, to make the generated candidate areas more accurate in the RPN network, and to make the classification effect better in the Fast-RCNN network.

Limited by the research time, the pedestrian detection algorithm research in this paper is not deep and perfect enough, and the system still needs to be improved and optimized in follow items:

(1). The problem of partial occlusion of pedestrian targets in detection scenes still exists. When the environment is complex and there are many detection targets,

occlusion is still easy to occur and cause target loss. In practical scenarios, occlusion problems occur more frequently. The existing methods can solve some local occlusion problems through different attempts, but the serious occlusion problems cannot be solved well.

(2). Human recognition of objects is a process. At the beginning, only have some memory of the basic information of the object, but don't know the category. Then I learn to know other category information, and next time I see the object, I can correctly recognize it. For pedestrian detection, labeling pedestrian targets is also a time-consuming and labor-intensive task, and it is meaningful to apply human learning mechanism to this field.

(3). Although compared with the traditional detection algorithm, Faster-RCNN has greatly improved both the detection speed and the detection accuracy. However, it still cannot achieve real-time detection. Although the recently emerged one-stage algorithm (SSD, yolo) does not improve the detection accuracy, it greatly improves the detection speed, making real-time monitoring become a reality, which will become the research direction of the author in the future.

7. Reference

[1]. Benenson, Rodrigo, et al. Ten Years of Pedestrian Detection, What Have We Learned.ECCV 2014 Workshops. Springer International Publishing, 2014:613-627

[2]. Bengio, Yoshua, A. Courville, and P. Vincent. Representation learning: a review and new perspectives. IEEE Transactions on PAMI 35.8(2013):1798-1828

[3]. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. ICLR, 201

[4]. M. Everingham, L. Van Gool, C. K. I.Williams, J.Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. IJCV, 2010.

[5]. R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

[6]. Girshick, R. Fast R-CNN. ICCV, 2015: 1440-1448.

[7]. K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. ECCV, 2014

[8]. Ren, S., He, K., Girshick, R., & Sun, J. Faster r-cnn: towards real-time object detection with region proposal networks. Computer Science, 2015

[9]. Uijlings, J. R. R., Sande, K. E. A. V. D., Gevers, T., & Smeulders, A. W. M. Selective search for object recognition. IJCV, 2013:104(2), 154-171.

[10]. Chatfield K, Simonyan K, Vedaldi A, et al. Return of the Devil in the Details: Delving Deep into Convolutional Nets[J]. Computer Science, 2014

[11]. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. CVPR, 2009

[12]. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. NIPS, 2012.

[13]. Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional Architecture for Fast Feature Embedding[J]. Eprint Arxiv, 2014:675-678

[14]. M. Everingham, L. Van Gool, C. K. I.Williams, J.Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. IJCV, 2010.

[15]. Uijlings J R R, Van De Sande K E A, Gevers T, et al. Selective search for object recognition[J].

[16]. Sande K E A V D, Uijlings J R R, Gevers T, et al. Segmentation as selective search for object recognition[C] IEEE International Conference on Computer Vision. IEEE, 2012:1879-1886.

[17]. Carreira.J, Sminchisescu C. Constrained parametric min-cuts for automatic object

[18]. segmentation[C]. Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.

[19]. Carreira J, Sminchisescu C. Cpmc: Automatic object segmentation using constrained parametric min-cuts[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(7): 1312-1328.

[20]. Arbeláez P, Pont-Tuset J, Barron J T, et al. Multiscale combinatorial grouping[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 328-335.

[21]. Pont-Tuset J, Arbelaez P, Barron J T, et al. Multiscale combinatorial grouping for image segmentation and object proposal generation[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(1): 128-140.

[22]. Cheng M M, Zhang Z, Lin W Y, et al. BING: Binarized normed gradients for objectness estimation at 300fps[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 3286-3293.

[23]. Zitnick C L, Dollár P. Edge boxes: Locating object proposals from edges[C]. European Conference on Computer Vision. Springer, Cham, 2014: 391-405.

[24]. Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.

[25]. He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[C]. European Conference on Computer Vision. Springer, Cham, 2014: 346-361.

[26]. Girshick R. Fast RCNN[C]. Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.

[27]. Ren S, He K, Girshick R, et al. Faster RCNN: Towards real-time object detection with region proposal networks[C]. Advances in neural information processing systems. 2015: 91-99.

[28]. Gidaris S, Komodakis N. Object detection via a multi-region and semantic segmentation-aware cnn model[C]. Proceedings of the IEEE International Conference on Computer Vision. 2015.

[29]. Edward Rosten, Reid Porter, Tom Drummond. Faster and Better: A Machine Learning Approach to Corner Detection[J]. IEEE transactions on pattern analysis and machine intelligence, 2008, 32(1):105-119.

[30]. Harris C. A combined corner and edge detector[J]. Proc Alvey Vision Conf, 1988.

[31]. Shi J, Tomasi C. Good features to track[C]. Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94. 1994 IEEE Computer Society Conference on. IEEE, 2002:593-600.

[32]. Calonder M, Lepetit V, Strecha C, et al. BRIEF: binary robust independent elementary features[C]. European Conference on Computer Vision. Springer-Verlag, 2010:778-792.

[33]. Bay H, Tuytelaars T, Gool L V. SURF: Speeded Up Robust Features[J]. Computer Vision & Image Understanding, 2006, 110(3):404-417.

[34]. Rublee E, Rabaud V, Konolige K, et al. ORB: An efficient alternative to SIFT or SURF[C].IEEE International Conference on Computer Vision. IEEE, 2011:2564-2571.

[35]. Felzenszwalb P, Mcallester D, Ramanan D. A discriminatively trained, multiscale, deformable part model[C]. Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008:1-8

Appendix

(1). SingleAnnotation.py

```
import os
import xml.dom.minidom as minidom
import numpy as np
import cPickle
import subprocess

def readSingleAnno(filename):
   with open(filename) as f:
      data = f.read()

   import re
   objs = re.findall('\(\d+, \d+\)[\s\-]+\(\d+, \d+\)',
data)

   num_objs = len(objs)

   boxes = np.zeros((num_objs, 4), dtype=np.uint16)
   gt_classes = np.zeros((num_objs), dtype=np.int32)
   overlaps = np.zeros((num_objs, 2), dtype=np.float32)

         # "Seg" area here is just the box area
   seg_areas = np.zeros((num_objs), dtype=np.float32)

      # Load object bounding boxes into a data frame.
   for ix, obj in enumerate(objs):
      # Make pixel indexes 0-based
     coor = re.findall('\d+', obj)
     x1 = float(coor[0])
     y1 = float(coor[1])
     x2 = float(coor[2])
     y2 = float(coor[3])
     boxes[ix, :] = [x1, y1, x2, y2]
     seg_areas[ix] = (x2 - x1 + 1) * (y2 - y1 + 1)
return boxes
```

(2). BatchAnnotations.py

```
import os
import numpy as np
import singleAnnotation as sann

#folder = "F:\INRIAPerson\Train\annotations"
#path = 'F:\INRIAPerson\Train\\annotations\\'
path = 'F:\INRIAPerson\Test\\annotations\\'
print path

imagelist = open('ImageList.txt', 'w')
annotations = open('annotations.txt', 'w')
for root, dirs, files in os.walk(path):
   for fn in files:
     #print (root + fn)
     imagelist.writelines(root + fn + '\n')
     boxes = sann.readSingleAnno(root + fn)
     #print len(boxes)
     annotations.writelines(root + fn + str())
                    #str(boxes[0][0]) + '\n')

   for i in range(0, len(boxes)):
      annotations.writelines(str(boxes[i][0]) + ' ' +
                             str(boxes[i][1]) + ' ' +
                             str(boxes[i][2]) + ' ' +
                             str(boxes[i][3]) + ' ')
   annotations.writelines('\n')

imagelist.close()
annotations.close()
```